UNIVERSITY OF CINCINNATI

Date: 6-Aug-2010

I, Tianyi Wang **,**

hereby submit this original work as part of the requirements for the degree of:

Doctor of Philosophy

in Industrial Engineering

It is entitled:

Trajectory Similarity Based Prediction for Remaining Useful Life Estimation

Student Signature: Tianyi Wang

This work and its defense approved by:

Committee Chair: Jay Lee, PhD
Jay Lee, PhD

Trajectory Similarity Based Prediction for Remaining Useful Life Estimation

A dissertation submitted to the

Graduate School

of the University of Cincinnati

in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

in the School of Dynamic Systems

of the College of Engineering and Applied Science

by

Tianyi Wang

University of Cincinnati

August 2010

Committee Chair: Jay Lee, Ph.D.

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Abstract

Trajectory Similarity Based Prediction for Remaining Useful Life Estimation (126 pp.)

The degradation process of a complex system may be affected by many unknown factors, such as unidentified fault modes, unmeasured operational conditions, engineering variance, environmental conditions, etc. These unknown factors not only complicate the degradation behaviors of the system, but also lower the quality of the collected data for modeling. Due to lack of knowledge and incomplete measurements, certain important context information (e.g. fault modes, operational conditions) of the collected data will be missing. Therefore historical data of the system with a large variety of degradation patterns will be mixed together. With such data, learning a global model for Remaining Useful Life (RUL) prediction becomes extremely hard. This has led us to look for advanced RUL prediction techniques beyond the traditional global models.

In this thesis, a novel RUL prediction method inspired by the Instance Based Learning methodology, called Trajectory Similarity Based Prediction (TSBP), is proposed. In TSBP, the historical instances of a system with life-time condition data and known failure time are used to create a library of degradation models. For a test instance of the same system whose RUL is to be estimated, similarity between it and each of the degradation models is evaluated by computing the minimal weighted Euclidean distance defined on two degradation trajectories. Based on the known failure time, each of the degradation models will produce one RUL estimate for the test instance. The final RUL estimate can then be obtained by aggregating the multiple RUL estimates using a density estimation method.

A case study using the turbofan engine degradation simulation data supplied by NASA Ames is provided to study the performance of TSBP. In this study, the TSBP method has demonstrated significant improvement in performance over a Neural Network based prediction method.

To my wife, Dandan

Acknowledgements

First I would like to express my sincere respect and gratitude to my advisor, Prof. Jay Lee. I thank him for his visionary guidance and persistent support to my research. I also appreciate him for his tireless advice to help me grow up as a strong person with rigorous altitude to work, breadth of knowledge, confidence, diligence and execution power.

I'm also grateful to Prof. Samuel Huang, Prof. Ernest Hall and Dr. Kai Goebel for serving as my doctoral committee. Prof. Huang and Prof. Hall used to teach me in the graduate classes that have contributed to an integral part of my knowledge in my current research field. Dr. Goebel has provided valuable advice to the experimental study conducted in this dissertation.

I would like to express my special appreciation to Mr. Patrick Brown for his help in improving my academic writing over the years. I'm thankful to the colleagues in the IMS Center, Dr. Masoud Ghaffari, Dr. Linxia Liao, Mr. Lei Yang, Mr. Edzel Lapira, Ms. Yan Chen, Mr. Hassan Al-Atat, Mr. Mohamed AbuAli, Mr. David Siegel, and the visiting scholars to the center, Dr. Shijin Wang, Dr. Jianbo Yu, Ms. Yang Yang, Mr. Fangji Wu, as well as many others who study and work in the center, for the frequent exchange of thought, inspiring brain-storming discussions and friendly support over the years.

Finally, many thanks go to my brother and my parents for their constant care to my study. Special thanks must go to my wife, Dandan, for her unconditional support and care day and night during my preparation of this dissertation.

Table of Contents

List of Tables

List of Figures

List of Symbols

Major symbols used in Chapter 3, 4, and 5.

$\mathbf{x}$	N-dimensional feature vector. $\mathbf{x} = (x_1, ..., x_N)^T$.
x_n	The n^{th} feature.
$\mathbf{x}_i$	The sample of feature vector at the i^{th} measurement cycle.
x_{ni}	The sample of the n^{th} feature at the i^{th} measurement cycle.
$\mathbf{z}$	M-dimensional PC vector. $\mathbf{z} = (z_1, ..., z_M)^T$.
z_m	The m^{th} PC.
$\mathbf{z}_i$	The sample of PC vector at the i^{th} measurement cycle.
z_{mi}	The sample of the m^{th} PC at the i^{th} measurement cycle.
t	The time variable
t_i	The time stamp of the i^{th} measurement cycle.
I	The index of the latest measurement cycle for an instance.
E	The index of the End-of-Life measurement cycle for an instance.
P	The index of the Start-of-Prediction cycle for an instance.
$EoUP$	The index of End-of-Useful-Prediction cycle for an instance.
X_I	The time series of $\mathbf{x}$ up to the latest measurement cycle. $X_I = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_I\}$.
Z_I	The time series of $\mathbf{z}$ up to the latest measurement cycle. $Z_I = \{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_I\}$.
r_I	The estimated RUL at measurement cycle I.
r_I^*	The ground-truth RUL at measurement cycle I. $r_I^* \equiv r_E - r_I$.
${}^l\square$	A left super script applied to any of the above symbols, indicating the symbol corresponding to the l^{th} training instance or degradation model.
L	The total number of training instances.
K	The total number of test instances.
N	The dimension of feature vector.
M	The dimension of PC vector. Usually $M < N$.
${}^lG$	The l^{th} degradation model extracted from the l^{th} training instance.
${}^l\mathbf{g}(t)$	The l^{th} degradation model function. ${}^l\mathbf{g}(t) = ({}^lg_1(t), ..., {}^lg_M(t))$.
${}^lD^2$	Squared distance to the l^{th} degradation model trajectory.
${}^lS$	Similarity to the l^{th} degradation model trajectory.

ε	Gaussian noise term in a regression model.
τ	The variable of time lag between a test instance and a degradation model ${}^{l}G$.
λ	The variable of time scaling (degradation acceleration) between a test instance and a degradation model ${}^{l}G$.
v_i	The weight for the i^{th} sample of a test instance in calculating mean Euclidean distance.
$K(\bullet, \bullet)$	A kernel function.
ρ	Kernel width, or Radial Basis Function spread.
$\mathbf{u}$	The vector of operational conditions.
$\mathbf{u}_i$	The sample vector of operational conditions for the i^{th} measurement cycle.
O_p	The p^{th} operating regime (subspace) of the system's operation space spanned from the operational condition vector $\mathbf{u}$.
C_p	The membership value that an operational condition $\mathbf{u}$ belongs to the the p^{th} operating regime.
$f_c(\mathbf{u})$	A (any) clustering algorithm that outputs membership values to all clusters for a given operational condition $\mathbf{u}$.
$h_{HA}^{(p)}$	The health assessment model for the p^{th} operating regime.
$\bar{\mathbf{x}}^{(p)}$	Mean of those feature samples collected under the p^{th} operating regime.
$\mathbf{s}^{(p)}$	Standard deviations of those feature samples collected under the p^{th} operating regime.
$\mathbf{y}$	The resultant feature after normalizing $\mathbf{x}$ under the corresponding operating regime.
$\{\underline{s}_i\}$	The resultant time series after smoothing a 1-D time series $\{s_i\}$ by a certain filtering or smoothing algorithm.
eSNR$(\bullet)$	The Empirical Signal/Noise Ratio computed from 1-D time series data.
$\tilde{\mathbf{y}}$	The resultant feature after applying variable weighting to the normalized feature $\mathbf{y}$
α	Percentage of RUL prediction error bound, e.g. 0.2.
i_α	The index of the first RUL prediction that satisfies the α-bound criteria.
ρ_{RBF}	Radial Basis Function Network spread parameter.

List of Acronyms

AI	Artificial Intelligence
AP	Rate of Acceptable Prediction
CBM	Condition-Based Maintenance
CBR	Case-Based Reasoning
CG	Convergence
CMAPSS	Commercial Modular Aero-Propulsion System Simulation
eSNR	Empirical Signal/Noise Ratio
FDI	Failure Detection and Isolation
HPC	High-Pressure Compressor
HPT	High-Pressure Turbine
IBL	Instance-Based Learning
LPC	Low-Pressure Compressor
LPT	Low-Pressure Turbine
MED-DA	Minimal Euclidean Distance with Degradation Acceleration
MED-TL	Minimal Euclidean Distance with Time Lag
MED-TL-DA	Minimal Euclidean Distance with Time Lag and Degradation Acceleration
MSE	Mean Squared Errors
NN	(Artificial) Neural Network
PCA	Principal Component Analysis
PC	Principal Component
PH	Prediction Horizon
PHM	Prognostics and Health Management
PoF	Physics of Failure
RA	Relative Accuracy
RBF	Radial Basis Function
RCM	Reliability-Centered Maintenance
RMS	Root Mean Square
RUL	Remaining Useful Life
RVM	Relevance Vector Machine
TSBP	Trajectory Similarity Based Prediction

1 Introduction

> "Prognostics and Health Management is a system engineering discipline focusing on detection, prediction, and management of the health and status of complex engineered systems." — the first International Conference on PHM, 2008

In the past decade, engineering system Prognostics and Health Management (PHM) has gained more and more attention in academia and industry. PHM has envisioned a new business trend that is elevated from the traditional Reliability-Centered Maintenance (RCM) and Condition-Based Maintenance (CBM) practices. Among the many subjects within the scope of PHM, prognostics is no doubt the most fundamental one. Its outcome builds the foundation of other PHM components, such as optimal scheduling of operation, maintenance, logistics, etc. A successful prognostics application has great impact on cost reduction, availability and safety assurance, and accomplishment of critical missions.

In general, prognostics can be referred to as the detection of failure precursors and predicting how much time remains before a likely failure [Schwabacher and Goebel, 2007], that is, the Remaining Useful Life (RUL). While the detection of failure precursors is mostly application dependent and requires background knowledge to the system, RUL prediction is relatively independent. The techniques for RUL prediction are mostly common to all prognostics applications, which will be the focus of this thesis.

1.1 Motivation

Being a rapidly developing research filed, the research on RUL prediction has embraced a vast number of techniques and algorithms that come from multiple research field, such as reliability engineering, regression analysis, time series modeling, artificial intelligence, etc. Most of the existing algorithms for RUL prediction learn a global prediction model from the data. These global models can be effective in applications that involve only simple systems or isolated components, where the system's degradation behavior can be well characterized by simple hypotheses or established knowledge. For complex systems, successful stories of prognostics are still rare. This can be attributed to two fundamental issues during the development and deployment of RUL prediction techniques:

- Lack of knowledge to the system's failure mechanisms and fault modes;
- Incomplete context information of the collected data.

For complex systems, it is costly, time-consuming, and probably infeasible to fully understand the system's dynamics and employ the first-principle models for prognostics. In such situations, data-driven techniques that rely on the discovery of failure precursors buried in the collected condition data become the only choice. When a data-driven model developed and tested under controlled experiments or lab conditions is going to be deployed in the field, it has to face the challenges brought up by the complexity of the real-world system.

Many real-world systems, such as machine tools, wind turbines or aircraft engines, consist of a large number of components, which, while worn out, may cause different degradation behaviors for the system and lead to different fault modes. When limited knowledge to the system is available and no effective diagnostic tools exist to support fault identification, those fault modes of the system that are not identified

will become an unknown factor that causes diversified degradation patterns in the collected condition data, which will only be perceived as noise or uncertainty for RUL modeling. In addition to fault modes, more unknown factors can be expected for a complex system. The system may operate under dynamically changing conditions; if not measured adequately, it will become another unknown factor that elevates data variation. Other potential unknown factors include engineering variance, environmental conditions, human factors, etc.

These unknown factors not only complicate the degradation behaviors of the system, but also lower the quality of the collected data for modeling. Due to lack of knowledge and incomplete measurements, certain important context information (e.g. fault modes, operational conditions) of the collected data will be missing. Historical cases of the system with large variety of degradation patterns and incomplete context information will be mixed together, which poses great challenge for RUL modeling. It becomes very hard for a global prediction model trained from such data to deliver good performance. This has led us to look for advanced RUL prediction techniques beyond the basic prediction methods.

One solution is to employ the common ensemble modeling techniques in machine learning, such as boosting, bagging, to enhance the model performance. These methods produce a complex prediction model consisting of multiple local models, whose internal mechanisms are hard to interpret. Another solution is to employ the Instance-Based Learning (IBL) or Case-Based Reasoning (CBR) methodology and develop models that are intrinsically built on a large number of historical cases. Due to the advancement of sensing and communication technologies, massive data collection from machines and equipment in the field, such as commercial aircraft engines or heavy-duty mining machines, becomes feasible. The abundant life-cycle condition data collected from multiple instances of the system has enabled the utilization of the

second solution, i.e. the instance-based RUL prediction approach.

The past research on IBL/CBR for engineering applications mainly focuses on diagnostics; the several applications on prognostics seldom utilize the information of degradation data as a means to evaluate instance similarity. Therefore there is still a research gap in utilizing IBL for prognostics applications. This has motivated the work in this thesis on the development of Trajectory Similarity Based Prediction (TSBP) approach, a novel RUL prediction approach based on the IBL methodology.

1.2 Objective and methodology

The objective of this research is to develop an effective RUL prediction methodology that can address the following issues in prognostics applications of complex engineering systems:

- Lack of knowledge of the system's failure mechanisms and fault modes;
- Incomplete context information of the collected data.

The research involves the following tasks:

- Establish the RUL prediction framework and methodology
- Find the best practices to tune model parameters
- Benchmark with other RUL prediction techniques through case studies.

Specifically in this thesis, the TSBP method is developed to fulfill the research goals. The method uses historical condition data from multiple training instances with known failure time to build a library of degradation models. For a test instance, similarity between it and each of the degradation models in the library are evaluated based on the distance metric defined for two degradation trajectories. Each degradation model will produce one RUL estimate for the test instance based on

the known failure time of the model. The multiple RUL estimates will be aggregated based on the similarity score to make the final RUL prediction.

1.3 Contributions and broader impacts

The key intellectual contributions of this thesis are highlighted below:

1. Developed an effective RUL prediction method that addresses multiple challenges in complex system prognostics;
2. Derived three similarity metrics between degradation trajectories, which enrich the IBL methodology in prognostics applications;
3. Developed a multi-regime data normalization method for data preprocessing, especially a variable weighting method applied as preparation to the regular Principal Component Analysis.

In addition, a solid case study is provided in this thesis to fully explore the strength and weakness of the developed methodology. TSBP has demonstrated effective prediction capability for complex system RUL estimation and noticeable improvement compared with the traditional Neural Network based prediction method.

The TSBP method developed in this thesis is expected to have a broader impact to industry. TSBP can be widely applied in engineering system prognostics applications where persistent data collection from a large number of identical machines or equipment are performed, such as for aircraft engines, heavy-duty mining trucks, wind farms and so on. Effective RUL predictions provided by the TSBP method will have great impact on the reduction of unplanned downtime and cost, safety assurance, and accomplishment of critical missions in such applications.

1.4 Thesis layout

The thesis will be organized as follows.

Chapter 2 discusses the basic issues and common techniques for prognostics and RUL prediction. Many RUL prediction techniques will be reviewed and summarized. Then performance evaluation methods and metrics for RUL prediction will be presented. At last, the common challenges in prognostics application will be summarized.

Chapter 3 brings forward the TSBP methodology, including the prognostics framework and key procedures. The assumptions made by the TSBP method will be discussed.

Chapter 4 discuss the data preparation method, which is a facilitating procedure under the TSBP prognostic framework. The focus will be put on the case that the data are collected under dynamic operational conditions. A multi-regime health assessment approach and a multi-regime data normalization approach are presented.

Chapter 5 gives a case study of TSBP on Turbofan engine degradation simulation data. The method to determine TSBP model parameters are presented. The algorithm performance is evaluated using a four-step hierarchical evaluation framework. At last, benchmarking results from the Neural Network approach is shown.

Chapter 6 summarizes the developed TSBP methodology and restates the contributions and broader impacts of the research work. Further more, a few comments on TSBP are made and the future works are laid out.

2 Issues and Techniques for RUL Prediction

In this chapter, the various issues, concept and terms related to prognostics and RUL prediction will be explained. The state-of-the-art technologies will be reviewed. The common challenges for RUL prediction will be discussed.

2.1 Prognostics and RUL prediction

Usually, *Prognostics* is defined as the detection of failure precursors, and the prediction of remaining useful life [Saxena et al., 2008a]. Detection of failure precursors can be considered as health assessment, the computation of the system's health states or health indices. RUL refers to the time left (from present) before a certain failure criteria is reached. It has several synonyms such as residual life, remnant life, time to failure, etc., from various research field. Based on its definition, prognostics in general requires two types of techniques: (a) the application-dependent techniques to detect failure precursors or to estimate the system's health state, and (b) the prediction techniques to predict RUL. While system health state estimation is mostly application dependent, RUL prediction is common in many prognostics applications. In this thesis, the focus will be put on RUL prediction techniques while the first type of techniques will be touched only when necessary.

RUL prediction is a type of event prediction that is related to but different from health state prediction. In many cases, RUL prediction needs to predict the health state far into the future until the failure criteria is reached. In some literature, the short-term prediction of the system's future health state without evaluating the failure criteria is also referred to as *Prognostics*. The predicted health state, however,

conveys only vague information regarding to the severity of the condition. It is not as actionable as time-to-failure for the purpose of predictive maintenance scheduling. The reason for this is easy to understand. If the progression characteristics of the health state is not fully understood, predictions on the health state may under- or over-estimate the actual severity of the system's health condition. As illustrated by Fig. 2.1, a system (a) whose current or near-term future state is far away from the failure threshold is not necessarily more critical than system (b) whose state is closer to the failure threshold, because system (b) may deteriorate with a slower rate towards the failure threshold. For example, an application of structure health prognostics employs crack length prediction techniques. The crack length shows the state of fault at a certain time point; as long as the predicted crack length does not cross the failure threshold yet, it is hard to decide whether one should repair or replace the structure at a certain future time point. RUL prediction, on the contrary, already taking into account the degradation progression pattern, provides a more straightforward health indicator (i.e. the RUL) that can be more easily utilized for decision making.

In addition, in case of unobservable or ambiguously-defined health state (e.g. a virtual health index without direct physical meaning. See also Section 2.4.2), the information conveyed by state predictions is even more vague. In fact, the maintenance paradigm based on predictions of the system's future state does not fundamentally change the traditional CBM paradigm. Therefore defining prognostics through RUL prediction is helpful to distinguish PHM from the traditional maintenance paradigms.

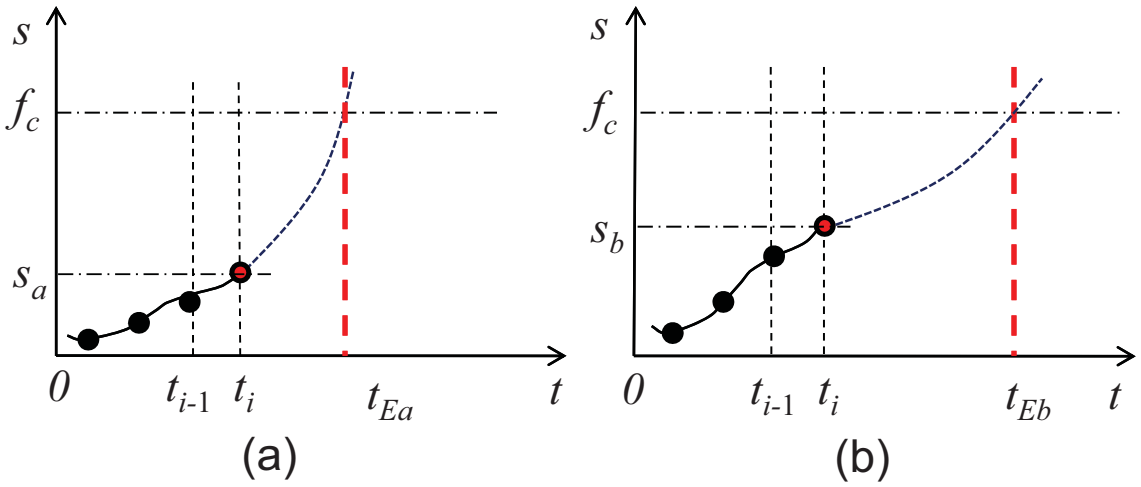

Figure 2.1: RUL prediction provides richer information about the system's health state. A state closer to the failure threshold (case b) does not necessarily imply higher criticality than a state farther away from the failure threshold (case a).

2.2 Failure and failure criteria

A definition of failure is critical for RUL prediction. In general, failure can be defined as the loss of ability to perform the required function. Depending on the needs of different applications, failure can be defined as

- A hard/physical failure, e.g. broken parts in the system, or,
- A soft failure, e.g. the system fails to meet the requirement of reliability level.

The failure time can be decided either

- Explicitly by a mathematically-defined failure criteria, e.g. a threshold, or,
- Implicitly based on the historical cases that are "'claimed"' to have failed.

In the first case, the failure criteria can also be defined in different ways. It can be defined as exact thresholds (as shown in Figure 2.1) on individual variables

or a function of them; or it can be defined as a probabilistic threshold based on the requirements of system reliability. The failure criteria can be given as a design specification; otherwise it has to be estimated from the historical cases of failures. Claim of failure can follow the First-Passage-Time [Whitmore, 1986] model as used in reliability analysis, i.e., a failure is claimed when the detected system's state passes the threshold for the first time; or it can be decided by multiple passages in consecutive measurements using, such as, Sequential Probability Ratio Test (SPRT) [Wald, 1945]. The failure criteria is required by the RUL prediction methods based on health state prediction discussed in Section 2.4.3 and 2.4.4.

In the second case, the failure time of the monitored system is not decided based on a single or multiple measurements or health state predictions; instead, it is estimated from the failure time of the historical failed cases of the system. The historical cases can be claimed to fail by a certain failure criteria not provided explicitly to RUL prediction (e.g. from the controller whose internal logics is unknown). In some situations, the failure time can be decided by subjective judgment. For example, the recorded time of system overhaul, or the time of a certain major maintenance action can be treated as the failure time. The implicitly provided failure criteria can be utilized by the many direct RUL estimation approaches discussed in Section 2.4.5.

RUL prediction always use the defined failure condition as reference. RUL prediction cannot be used to estimate the extra life beyond the failure reference. For instance, corrective maintenance is always applied to a system based on a certain reliability requirement and the system has never operated beyond that reliability threshold. If the point of corrective maintenance is defined as the failure point (a soft failure), then RUL prediction will not be able to provide a reliable life estimate with reference to a hard failure beyond the reliability threshold.

2.3 Scope of RUL modeling

RUL prediction is meaningful only for those engineering systems with evolving degradation behaviors. Usually it does not apply to the system with stochastic failures such as certain electronic components. For many engineering systems, especially mechanical systems, the degradation process is irreversible unless the condition is recovered by effective maintenance actions. The irreversible degradation process of a system does not necessarily imply monotonic progression of the observed features of the system though. For instance, RMS of a bearing's vibration amplitude may not be monotonic increasing during wear.

RUL modeling focuses the long-term, slow characteristics of the degradation process. Thus the local, short-term behavior of the system can be treated as disturbance. Catastrophic failures caused by unpredictable events will not be considered. Infant failures (the system fails premature during run-in period), though important for reliability analysis, are usually not modeled for RUL prediction purposes either.

Usually RUL modeling does not model the effect of major preventive maintenance actions within the prediction horizon. Preventive maintenance may change the degradation progression of a system dramatically by recovering the system's health or performance to a new level, which is usually unpredictable. Unless a prediction model is specially designed for this application, the effect of preventive maintenance will be treated as noise in the degradation process.

2.4 Review of RUL prediction techniques

2.4.1 Classification of RUL prediction techniques

Being a rapidly developing subject, the research on RUL prediction has employed a vast number of techniques in multiple research areas, such as regression analysis, time series forecasting, statistical survival analysis, Artificial Intelligence, etc. These techniques can be classified into three categories, representing three levels of prognostics (a classification method adapted from Vachtsevanos et al. [2006]): *model-based approaches*, *data-driven approaches* and *experience-based approaches*, as shown in Figure 2.2. The approaches towards the top of the pyramids incur higher cost in development, deliver higher accuracy and is more application specific.

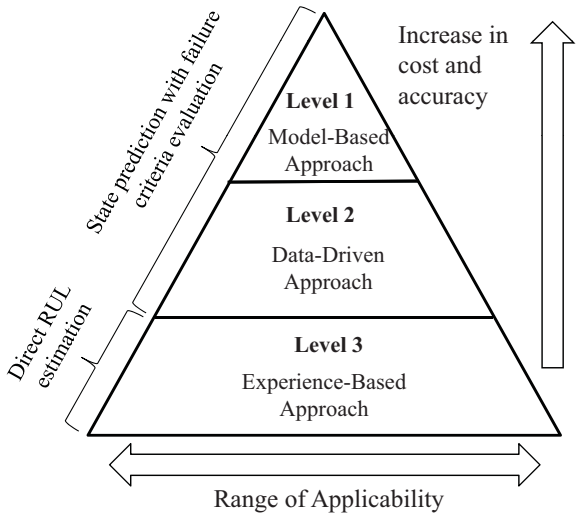

Figure 2.2: Three-level classification of RUL prediction techniques (adapted from Vachtsevanos et al. [2006, p.289])

Both the model-based approaches and the data-driven approaches rely on state prediction followed by failure criteria evaluation, which includes two essential

procedures: 1) estimating the system's health state (health index, degradation stage, etc.), and 2) predicting or extrapolating the system's state up to the time when the failure criteria is satisfied. The difference is that the model-based approaches make predictions through physics models or system models, while the data-driven approaches make predictions through data models learned from the time series of states through regression/trend analysis or stochastic process modeling. These approaches require an explicit failure criteria provided *a priori*. If not, it has to be first estimated from the given data of failure cases.

The experience-based approaches estimate RUL directly by modeling the relations between the states, the current life and the failure time, without an explicit failure criteria.

Common RUL prediction approaches falling into these three categories are summarized in Table 2.1.

2.4.2 State estimation

State estimation is required by many RUL prediction approaches as a preparation stage, except for those approaches relying on feature-level predictions. Here, system state is characterized by a continuous health indicator/index or discrete degradation stages. The health indicator is usually a continuous-value quantity defined with one of the following methods:

Physical health indicator is directly defined by a physical parameter of the system, such as crack length in a gear, or the vibration amplitude of a shaft. The threshold can be decided by the design specification.

Probabilistic health indicator is defined by a certain probability, e.g. the system's reliability defined in the Proportional Hazard Model [Banjevic and Jardine, 2006], or the probability of the current system being in a healthy condition evaluated by

Table 2.1: Summary of RUL prediction approaches

Category	Algorithms/methods	Characteristics	

Level 1: Model-based approach	• Physics-of-Failure modeling
	• Model-based FDI
	• Hybrid models
	  • (Extended) Kalman filtering
	  • Particle filtering
		• State prediction achieved through physics or system models	State prediction with failure criteria evaluation

Level 2: Data-driven approach	• Linear/nonlinear regression • ARMA/ARIMA • AI methods (Neural Networks, Fuzzy Logic, ANFIS, etc.) • Hidden Markov Model • Gaussian process regression • SVM/RVM	• State prediction achieved through regression or stochastic process modeling	

Level 3: Experience-based approach	• AI methods
	• Reliability models
	  • Weibull models
	  • Proportional Hazard Model
	  • Proportional Integrated Model
	• Stochastic filtering
	• Instance based learning
		• Direct modeling of the relations between the current life, measured conditions and the failure time	Direct RUL estimation from failure histories

logistic regression models [Yan et al., 2004], or the T-square statistics [MacGregor and Kourti, 1995] on process novelty. The value of this health indicator is between 0 and 1, with 1 meaning the best (or the worst, depending on the definition) state of the system. The threshold of this type of health indicator can be set by a statistical confidence level.

Mathematical health indicator is defined by a variable with only mathematical meaning, such as a certain distance metric, e.g. Mahalanobis distance, Minimal Quantization Error in a Self-organizing Map (e.g. in Huang et al. [2007]), L2 distance between distributions (e.g. in Liu et al. [2007]); it can be virtually any scalar value transformed from the multi-dimensional feature space. The residual generated in Model-based FDI (see also Section 2.4.3.2) can be treated as a mathematical health indicator. Similarly, the residual generated by the Multivariate State Estimation Techniques (MSET) [Cheng and Pecht, 2007] is another mathematical health indicator. The threshold for this type of health indicator have to be learned from training data set, or specified heuristically.

Discrete degradation stages of a system are required by certain modeling techniques such as the Hidden Markov Models (see Section 2.4.4.5). They may be defined empirically, mathematically or based on physics. They can also be obtained by discretization of the continuous-value health indicators.

2.4.3 Model-based approaches

The model-based approach is a modeling paradigm for system state estimation and prediction. Although first-principle models are most frequently used in these methods to model the mechanism of the system's state progression, they are not the only models that can fit into the model-based modeling paradigm. A certain data-driven, mathematical models can also be used in this paradigm.

2.4.3.1 Modeling using Physics-of-Failure

Physics of Failure (PoF) modeling is usually applied at material level or component level, and derived from wear or failure mechanism. A typical PoF model for mechanical components is the fault propagation models, such as cracks or spalls propagation [Li et al., 1999, 2000; Marble and Morton, 2006; Ray and Tangirala, 1996]. Fault propagation are usually affected by the loading (operational) conditions of the components. Therefore fault propagation models usually incorporate the loading conditions as inputs and compute cumulative damage over time. The future loading condition, however, is sometimes an uncertain factor under dynamic operational conditions. The common way to deal with this uncertainty is to assume a certain loading pattern, e.g. constant loading, for the future. Multiple loading scenarios can be applied to evaluate the component's RUL with different usage behaviors.

Many characteristic parameters for PoF models have to be identified using experimental data. Backed up by the first principles regarding to relations of the measurement data (such as vibration) and experimental settings (such as loading), it is possible to use a fewer number of experiments to identify the characteristics of the system comparing to what's needed for data driven methods. However, due to modeling assumptions, errors and unforeseeable uncertainty in the application, the model may not be as accurate as it is developed in the experiments. Therefore the PoF models are usually integrated with on-line parameter updating methods based on the measured condition data at run time; this essentially become a hybrid modeling method, which will be addressed later.

2.4.3.2 Model-based FDI

Model-based Fault Detection and Isolation (FDI) is a subfield of control engineering and has been studied for more than 40 years. Model-based FDI requires

the mathematical model of the physical system that is under consideration; it makes FDI decisions based on the residual generated between the physical system and the mathematical model. Although there are more than on type of model-based FDI scheme, they all end up with state estimation or parameter estimation [Frank, 1990]; from health prognostics point of view, the estimated state or parameters can be used as the health indicators, either physical or mathematical ones.

Frank [1990] summarized four types of model-based FDI schemes. The *parity space* approach, also called analytical redundancy, uses the model to simulate system output with given inputs, and use the error between the physical system output and the simulation output as the fault indicator. The *dedicated observer* approach is used to estimate the hidden state of the system using both the inputs and outputs of the physical system; the estimated state can be a health indicator with clear physical meaning. The *fault detection filter* (FDF) approach is used to model the impact of particular faults to the system's input-output behavior; the FDF models can then be used to detect and isolate a fault when the system behavior can be explained by the particular FDF model. The *parameter identification* approach tries to learn the system parameters from the input/output of the physical system and use the change of system parameters as health indicators.

Traditionally, model-based FDI approaches do not model fault progression inherently; in such cases the generated states or health indicators has to be combined with other prediction tools to achieve prognostics. Recently, the model-based FDI modeling scheme is also employed under prognostics settings, where the mathematical models direct model the fault propagation behavior of the system; with the *dedicated observer* modeling scheme, the system state can be estimated recursively into the future for failure threshold evaluation. These types of FDI models usually integrate the recursive filtering techniques, such as Kalman filter, Extended Kalman filter and

Particle filter, to achieve robust state estimation – this leads to the so-called hybrid modeling approach (see the next section).

The modeling scheme of model-based FDI has also been extended for data-driven modeling. Instead of deriving the models from first principles, generic black-box models, such as Nonlinear Autoregressive Exogenous (NARX) models [Leontaritis and Billings, 1985], Artificial Neural Network, etc., can also be used to model the input-output relations of the system. The parameters of these models, similar to the first-principle models, can be identified from the training data.

2.4.3.3 Hybrid models

Recursive Bayesian estimator is an important approach to integrate state-space models of the system with the sensor measurement data and enables the so-called hybrid models for prognostics. Recursive Bayesian estimators have two common procedures at each iteration: predict and update. Let $\mathbf{s}_i$ be the unobservable state that is assumed to have Markovian properties, i.e. the current state $\mathbf{s}_i$ is only dependent on the immediate leading state $\mathbf{s}_{i-1}$ but independent from all other earlier states $\mathbf{s}_{i-2}, \mathbf{s}_{i-3}, ..., \mathbf{s}_0$. Let $\mathbf{o}_i$ be the observation variable that is assumed to be only dependent on the current state $\mathbf{s}_i$ but independent from all other states and observations. Then a generic Bayesian filter will have the following form for recursive prediction and update:

$$\text{Prediction: } p(s_i|o_{i-1}) = \int p(s_i|s_{i-1})p(s_{i-1}|o_{i-1})ds_{i-1} \tag{2.1}$$

$$\text{Update: } p(s_i|o_i) = \frac{p(o_i|s_i)p(s_i|o_{i-1})}{p(o_i|o_{i-1})} = \alpha \cdot p(o_i|s_i)p(s_i|o_{i-1}) \tag{2.2}$$

where the denominator $p(o_i|o_{i-1})$ is used for normalization purposes and thus can be replace by a coefficient α. The transition probability $p(s_i|s_{i-1})$ and observation probability $p(o_i|s_i)$ are priors decided by the state-space model of the system, which

can be derived from physics or learned from the data.

Two important variants of recursive Bayesian estimators are Kalman Filter [Welch and Bishop, 1995] and Particle Filter [Arulampalam et al., 2002]. Kalman filter is an optimal recursive Bayesian estimator for linear state-space model with Gaussian noise. For non-linear state-space models, linearization around the current point can be made and this leads to the Extended Kalman Filter. Ray and Tangirala [1996] presented a nonlinear stochastic model of fatigue crack dynamics and used Extended Kalman Filter for online fault prognosis. Note that the EKF is not an optimal estimator any more; if the initial estimate of the state is significantly off-target, or if the process is modeled incorrectly, the filter may quickly diverge, owing to its linearization [Saha et al., 2009]. For non-linear state-space model with non Gaussian noise, the Particle Filter provides a better solution. Particle Filter, a recursive Bayesian estimator based on sequential Monte Carlo (MC) simulations [Saha et al., 2009], models the probability density functions using a set of discrete points. Orchard and Vachtsevanos [2009] presented a particle filtering framework for fault prognosis.

2.4.4 Data-driven approaches

The data-driven approaches here refer to those methods that models the degradation behavior from time series of the system state (or the raw measurement data, the computed features, etc.), and make predictions based on the learned models. Most data-driven approaches do not model system inputs; they only model the behavioral measurement data. The system's future operational profile, i.e. the inputs, is assumed to be constant or at least consistent with the past. The data-driven approaches discussed here predict the system's future state recursively until reaching the failure criteria; in this way, the RUL can be obtained.

2.4.4.1 Linear/non-linear regression

The most basic method to make predictions from a time series is to fit a parametric regression model from the historical data and then extrapolate the model to the future time. Consider a 1-D time series $(T, X) = \{(t_i, x_i)|i = 1, ..., I\}$ generated from random variable x with time stamp t. A simple linear model can be expressed as

$$x = \beta_0 + \beta_1 \cdot t + \varepsilon \tag{2.3}$$

where (β_0, β_1) are model parameters to be found, and ε is the noise term. A generalized linear model has the following form

$$x = \beta_0 + \sum_{p=1}^{P} \beta_p \phi_p(t) + \varepsilon \tag{2.4}$$

where $\phi_p(t)$ can be any function of t, representing P basis functions of the linear model. For example, with $P = 2$, $\phi_1(t) = t$ and $\phi_2(t) = t^2$, Eq.(2.4) gives a second-order polynomial model

$$x = \beta_0 + \beta_1 \cdot t + \beta_2 \cdot t^2 + \varepsilon$$

Similarly, certain non-linear parametric models can also be used to fit the time series data. For example, an exponential model with four parameters has the following form

$$x = \beta_0 + \beta_1 \cdot \exp(\beta_2 \cdot t + \beta_3) + \varepsilon \tag{2.5}$$

Many techniques, such as Least Square, Maximum likelihood, etc., can be used to estimate the model parameters from the training samples (T, X). Once the parameters are obtained, prediction is as simple as evaluating the model response with a future time t as input. The challenge of the simple regression method is to find an appropriate form of parametric model that can best explain the degradation behavior in the time series. In practice, the best model form can be discovered from

historical failure cases with run-to-failure data, where a complete degradation pattern can be observed. However, the real-world systems' degradation pattern seldom follow the simple form of parametric models.

2.4.4.2 ARMA/ARIMA

The Autoregressive Moving Average (ARMA) model is a variant of the Box-Jenkins models [Box et al., 1994] that is capable of modeling autocorrelated time series data. For a time series $(T, X) = \{(t_i, x_i)|i = 1, ..., I\}$ with uniform timestamps (i.e. $\Delta t = t_{i+1} - t_i$ is constant), the ARMA model with order (P, Q) takes the following form

$$x_i = -\sum_{p=1}^{P} \varphi_p x_{i-p} + \sum_{q=1}^{Q} \theta_q \varepsilon_{i-q} + \varepsilon_i \tag{2.6}$$

where $\varphi_p x_{i-p}$ are P autoregressive terms and $\theta_q \varepsilon_{i-q}$ are Q moving average terms. As we can see that the ARMA model consists of two parts: an autoregressive (AR) part that addresses the observation values, and a moving average (MA) part that addresses the noises in the observations. The model without the MA terms is called an AR model. An ARMA model has less modeling error than an AR model due to the added MA terms.

A major limitation for ARMA models in prognostics applications is that they assume the time series to be stationary. For non-stationary signals, the Autoregressive Integrated Moving Average (ARIMA) models, another variant of the Box-Jenkins models, are more appropriate.

ARMA/ARIMA models support only one-step-ahead prediction. For long term prediction, the model has to be evaluated recursively, using the predicted value from previous steps as known inputs for a new prediction step. Note that ARMA/ARIMA are univariate models, therefore their usage in prognostics application is usually to

extrapolate a single health indicator. For instance, Yan et al. [2004] used ARMA model to extrapolate the health-index time series generated from a logistic-regression based health assessment model.

2.4.4.3 Artificial Neural Networks

The Artificial Neural Networks (ANN), or Neural Networks (NN) in short, can be treated as a multi-input-multi-output nonlinear blackbox function approximator, $\mathbf{y} = f_{NN}(\mathbf{x}; \Psi)$, where Ψ is the parameter vector. NN has the flexible to play different roles in prognostics applications, such as state estimation, state prediction or directly RUL modeling (see Section 2.4.5), depending on how the inputs and outputs are constructed.

When used for state estimation, the inputs to NN are multivariate measurements/features $\mathbf{x}$ and the output is the estimated 1-D state s:

$$s = f_{NN}(\mathbf{x}; \Psi) \tag{2.7}$$

When used for state prediction, the inputs to NN are consecutive states in the past and the output is the future states to be predicted:

$$(x_{i+1}, ..., x_{i+Q}) = f_{NN}(x_{i-P}, x_{i-P+1}, ..., x_{i-1}; \Psi) \tag{2.8}$$

When used for direct RUL estimation, the inputs to NN are consecutive states in the history and the output is the expected RUL:

$$r_i = f_{NN}(x_{i-P+1}, x_{i-P+2}, ..., x_i; \Psi) \tag{2.9}$$

All three scenarios have been reported in literature for prognostics applications. [Wang and Vachtsevanos, 2001] used wavelet neural networks (WNN) to evaluated crack from features, and then used dynamic wavelet neural networks (DWNN) to predict the fault propagation process and estimate RUL. Yam et al. [2001] applied

a recurrent neural network (RNN) for predicting the machine condition trend and Heimes [2008] applied RNN for aircraft engine RUL estimation. From the modeling point of view, a feed-forward NN is a special case of non-linear AR models, whereas an RNN, with feedback connections, is a non-linear AR moving average model [Wang et al., 2004]. Similar to the advantages of ARMA models over AR models, RNN as a predictor has advantages over the feed-forward NN, as verified by Tse and Atherton [1999] through simulation and practical tests.

2.4.4.4 Fuzzy Logic systems

The Fuzzy Logic (FL) systems provide another method for nonlinear function approximation. The difference from the NN approach lies in that FL gives more transparency to the mechanisms behind function approximation by using linguistic rules to integrate simple local models into a highly nonlinear model. A frequently-used FL model for function approximation or regression is the Takagi-Sugeno model [Takagi and Sugeno, 1985]. Similar to the use of NN, the input-output of the Takagi-Sugeno model can be constructed differently so that the model can be used to predict future states or the RUL directly.

Rule generation in a FL system usually requires expert knowledge. This is an advantage in that it can incorporate knowledge to guide model creation; however it may be also a disadvantage if the system is too complex to design rules one by one. This disadvantage can be overcome by the Neural-Fuzzy (NF) method, such as the Adaptive Neural Fuzzy Inference System (ANFIS) [Jang, 1993], which utilizes NN to learn the fuzzy system structure and rules. Of course, the use of NF method will make the FL system closer to a blackbox approach, i.e. NN. Wang et al. [2004] applied both NF and RNN to gear condition prediction and reported that NF can achieve better prediction accuracy than RNN. Chinnam and Baruah [2004] presented

a NF approach to estimate RUL for the situation where no failure data and no specific failure definition are available, but instead, domain experts are available.

2.4.4.5 Hidden Markov Models

The Hidden Markov Model (HMM) is a stochastic model consisting of unobservable states that have Markov properties and observations that are dependent only on the current state but independent from other observations. The HMM has similar form to the recursive Bayesian estimator discussed in Section 2.4.3.3; the difference is that the states in a HMM are usually discrete and the state-space model behind HMM has a probabilistic form rather than derived from physics. The HMMs has been primarily used for sequence clustering and classification for applications such as speech recognition [Rabiner, 1989] or dynamic system fault detection [SMYTH, 1994]. Due to the state-estimation capability of HMM, it has also been used for prognostics applications. For example, Baruah and Chinnam [2005] applied HMM for both fault diagnostics and prognostics of cutting tool condition in machining processes. Chinnam and Baruah [2009] proposed an autonomous diagnostics and prognostics method using competitive learning-driven HMM-based clustering in machining processes. When used for prognostics, especially for RUL estimation, the assumption of Markovian property is not always realistic because the HMM does not model temporal information about state transition. Dong and He [2007] proposed a segmental hidden semi-Markov model (HSMM)-based prognostics framework to address this issue. An HSMM allows modeling the time duration of the hidden states and therefore is capable of prognostics [Dong and He, 2007].

2.4.4.6 Other approaches

Relevance Vector Machine (RVM) [Tipping, 2001] is a sparse kernel method. RVM has an identical function form as a Support Vector Machine (SVM), but use a Bayesian inference approach to obtain solutions for regression and classification. Considering the generalized linear model given by Eq.2.4, both RVM and SVM can be expressed with the following form using kernel basis functions:

$$y(\mathbf{x}) = \sum_{i=1}^{I} K(\mathbf{x}, \mathbf{x}_i) w_i + w_0 \tag{2.10}$$

where $K(\bullet, \bullet)$ is the kernel function, w_i are weights and I is the number of training samples. Sparsity of RVM or SVM is achieved because the majority of the weights w_i will be zeros or close to zero once the model is learned from training; the non-zero weights are corresponding to the training samples that are called the relevance/support vectors. Then, SVM can make predictions using a deterministic approach while RVM can provide probabilistic outputs. One of the advantages of RVM over SVM is that RVM typically utilizes dramatically fewer basis functions than a comparable SVM [Tipping, 2001].

Gaussian Process Regression (GPR) [Williams, 1998] is another kernel method with Bayesian treatment for regression. While Gaussian distribution is over vectors, Gaussian process is over functions. A Gaussian Process is fully specified by the mean function $m(x)$ and covariance function $k(x, x')$. Therefore a function under GPR context can be considered to have a Gaussian process distribution with mean function $m(x)$ and covariance function k [Rasmussen, 2004]. It was shown that Gaussian processes models are equivalent to the NN with one hidden layer and an infinite number of hidden neurons [Neal, 1996].

Three prediction methods, GPR, RVM and NN, were used by Goebel et al. [2008] for RUL prediction of a rotating equipment in a test stand of an aerospace setting. It

concluded that NN performance varies with the choice of training data and also with its architecture, while RVM and GPR have the advantages to provide probability distribution for uncertainty management.

Wegerich [2004] presented a similarity-based modeling method derived from Multivariate State Estimation Technique (MSET) [Cheng and Pecht, 2007], which make predictions by averaging the training data based on a similarity measure. Liu et al. [2007] presented another formulation of similarity-based prediction, which make predictions by comparing signatures from any two degradation processes using measures of similarity that forms a Match Matrix. The method can effectively include large amounts of historical information into the prediction of the current degradation process.

2.4.5 Experience-based approach

This category of RUL prediction methods directly model the relation between the measurement data and RUL from historical failure cases. These models can predict or infer RUL directly without the need to evaluate the failure criteria because no predictions of system states will be made. The methods in this category include AI methods, reliability analysis methods, stochastic filtering method, and instance-based methods.

2.4.5.1 AI methods

Many generic nonlinear regression models or function approximators can be configured to make direct RUL estimation, as long as the model output/response variable are chosen as RUL directly. For example, [Tian and Zuo, 2009; Tian et al., 2009] employed a NN model for RUL prediction, with the condition data and unit age as inputs, and the RUL as output.

2.4.5.2 Reliability/survival analysis

The traditional reliability/survival analysis methods statistically model failure rate, reliability and failure time distributions from historical data. These methods can compute RUL distributions solely based on age, such as the Weibull models, or based on both age and condition data, such as the Proportional Hazard Models.

The Proportional Hazard Model (PHM) [1], since Cox's pioneering paper in 1972 [Cox, 1972], has been the most prevalent approach for survival analysis. It has been extensively used for reliability-centered maintenance and condition based maintenance practices. In order to incorporate condition monitoring data, a time-dependent PHM is required, whose hazard/failure-rate function is given as

$$h(t) = h_0(t)\exp\left(\beta_1 x_1(t) + ... + \beta_p x_p(t)\right) \tag{2.11}$$

where $h_0(t)$ is the baseline hazard function, $x_1(t), ..., x_p(t)$ are time-dependent covariates (the condition data) and $\beta_1, ..., \beta_p$ are coefficients. A frequently used baseline hazard function is the Weibull hazard function, which is the hazard function of the Weibull distribution. PHM with Weibull baseline hazard is called Weibull PHM, which is frequently used for optimal component replacement decisions in condition based maintenance [Jardine et al., 1987; Vlok et al., 2002]. Weibull PHM, however, cannot make RUL prediction by itself without additional predictions to the time-dependent covariates $x_1(t), ..., x_p(t)$. Banjevic and Jardine [2006] used a joint model of PHM and Markov to achieve RUL estimation, where the covariates are modeled by a Markov process.

Proportional Intensity Model (PIM) is an extension of PHM, which adopts a stochastic process setting and assumes a similar form to the intensity function of the

[1] The acronym PHM used in this section should not be confused with term of Prognostics and Health Management (PHM) used before.

stochastic process [Jardine et al., 2006]. Vlok et al. [2004] applied PIM with covariate extrapolation to estimate bearing residual life.

Note that PHM and PIM models use only the latest condition data to compute the failure time distribution, which makes the method subject to noise in the condition data.

2.4.5.3 Stochastic filtering

The stochastic filtering method [Wang and Christer, 2000; Wang et al., 2000] is a recursive Bayesian estimator (see Section 2.4.3.3) designed for direct RUL estimation. In this method, RUL is treated as a special state of the system and will be estimated and updated recursively from the collected condition data. Wang [2002] used the stochastic filtering method to predict the RUL of rolling element bearings given monitored condition information to date. Carr and Wang [2008] compared the stochastic filter with Weibull PHM for RUL estimation using oil-based condition monitoring data set. It shows that stochastic filtering approach provides better predictions than PHM in terms of MSE prediction errors.

2.4.5.4 Instance based learning

Instance Based Learning (IBL) provides a non-parametric solution for pattern analysis and prediction, which allow the hypothesis complexity to grow with the data. On the contrary, the traditional prediction methods assume a particular restricted family of models, which often oversimplifies what's happening in the real world [Russell and Norvig, 2003, p.733]. Typical IBL algorithms includes the k-nearest-neighbor method, Kernel Regression [Russell and Norvig, 2003], Locally Weighted Regression [Mitchell, 1997], and so on. These algorithms, however, represent instance as a vector in an n-dimensional Euclidean space. For the application of diagnostics

and prognostics, a richer instance representation is usually required – this has led to the technique of Case Based Reasoning (CBR) [Aamodt and Plaza, 1994], an extension to IBL, which relies on the philosophy of "similar cases produce similar outputs" for problem solving.

Past applications of IBL or CBR (e.g. Saxena et al. [2005]) has mainly focused on fleet-based diagnostics problems, with only a few cases addressing RUL estimation. Bonissone et al. [2005] used IBL for locomotive RUL estimation, where the instance was associated with a number of attributes such as the usage and maintenance history, and a Fuzzy instance model was proposed to evaluated instance similarity in the n-dimensional attribute space. Xue et al. [2008] employed a similar approach for aircraft engine RUL estimation. In both applications, the RULs of the training instances were estimated by a variety of methods such as heuristics or statistics, and then the RULs given by similar training instances were aggregated using Kernel Regression to make the final RUL estimate. Gebraeel et al. [2004] proposed a neural network approach for bearing residue life prediction, which had the nature of IBL methods too. In this application, features extracted from vibration signals of multiple bearing instances were first used to train multiple neural networks to establish the relation between the features and the bearing's usage time; then the neural networks were used to evaluate similarity between a test instance and the training instances. Huang et al. [2007] employed a similar approach for bearing RUL estimation, with the difference of improved feature extraction method. Recently, Zio and Maio [2010] presented a similarity-based approach for prognostics using Fuzzy point-wise similarity defined for degradation trajectory data. This method followed the same thinking as the method proposed in the earlier publication [Wang et al., 2008] and the TSBP method to be presented in this thesis; the difference mainly lies on the definition of similarity and the treatment method for model aggregation.

Generally speaking, the IBL approach for RUL estimation involves three essential tasks: instance retrieval, prediction through local models and aggregation of local predictions. Instance retrieval employs a certain definition of similarity (or distance) between the test and the training instances to generate weights for each training instances. Local instance models are required for each of the training instances to produce local RUL predictions. It is common to use the actual life of the training instance as the prediction of the test instance's life. Finally, the local predictions will be aggregated to obtain the final RUL estimate using weighted sum of the local predictions, where the weight is a function of the similarity score obtained during instance retrieval. The current usage of IBL for RUL estimation, however, has a limitation. During instance retrieval, the similarity between instances is defined either using a number of attributes of the instance [Bonissone et al., 2005; Xue et al., 2008], or using the feature vector at the latest measurement cycle [Gebraeel et al., 2004; Huang et al., 2007], or using a fixed number of past measurements [Zio and Maio, 2010]. Therefore the information contained in the up-to-date condition data is not fully utilized to decide instance similarity. This issue will be addressed by the TSBP approach presented in this thesis.

Note that IBL is a so-called "lazy learning" approach, which defers the decision of how to generalize beyond the training data until a query instance is provided [Mitchell, 1997]. Therefore the computational load during model training can be light but that during model evaluation can be much higher.

2.5 Methods of performance evaluation

Currently there's no widely acceptable performance evaluation method for prognostics applications. A large number of performance metrics are available, but the selection of them are usually applications specific, and sometimes subjective.

Some of the performance metrics are derived from other prediction-related subject, such as Mean Squared Error (MSE) from regression analysis. These metrics, however, may be incapable to address the special needs of prognostics. Recently years, some attempts have been made to develop a generic performance evaluation framework for prognostics applications. In this section, some frequent used traditional performance metrics as well as the performance evaluation framework proposed by Saxena et al. [2010] will be discussed.

2.5.1 Notations

For each test instance, an RUL r_i can be predicted with historical measurement data provided up to the time t_i. The true RUL at t_i is noted as r_i^*. During evaluation, the historical measurement data can be provided incrementally so that multiple RUL predictions can be made. These predictions will form a time series of predictions, $R = \{r_i|t_i\}$. Usually predictions will not be triggered when only a few measurements are available; the earliest time to start prediction is noted as time t_P with time index P. Also, predictions too close to the end of life of the system lose the practical value and thus will not be evaluated either; the last prediction is called End of Useful Prediction (term adopted from Saxena et al. [2010]) and the corresponding time is noted as t_{EoUP}. Therefore the valid range of time series $R = \{r_i|t_i\}$ is constrained by $t_P \leq t_i \leq t_{EoUP}$ or $P \leq i \leq EoUP$.

In many cases, multiple test instances will be used for performance evaluation. For the k^{th} out of K test instances, the predictions time series is noted as ${}^kR = \{{}^kr_i|t_i, {}^kt_P \leq t_i \leq {}^kt_{EoUP}\}$. The corresponding ground-truth RUL at each time stamp is noted as ${}^kr_i^*$.

Usually prediction performance is evaluated based on the errors between the predictions and the ground truth. The error is noted as ${}^k\Delta_i = {}^kr_i - {}^kr_i^*$.

2.5.2 Traditional performance metrics for predictions

Traditional performance metrics are functions of the prediction errors. These metrics treat each single RUL prediction independently, even though many predictions are made for the same instance but only at different time. Therefore all the predictions from K test instances will be combined to form a new set of predictions $\Omega = \{(r_j, r_j^*)\}$, eliminating the information of instances and timestamps. The total number of predictions in the set is $J = |\Omega| = \sum_{k=1}^{K} |^k R|$. Prediction errors is noted as $\Delta_j = r_j - r_j^*, j = 1, ..., J$. Frequent used performance metrics can be computed as follows.

2.5.2.1 Accuracy based metrics

Accuracy based metrics evaluate how close the prediction is to the ground truth.

- Bias:

$$\text{Bias} = \frac{1}{J} \sum_{j=1}^{J} \Delta_j$$

- Root Mean Squared Error:

$$\text{RMSE} = \sqrt{\frac{1}{J} \sum_{j=1}^{J} \Delta_j^2}$$

- Mean Absolute Error:

$$\text{MAE} = \frac{1}{J} \sum_{j=1}^{J} |\Delta_j|$$

- Symmetric Mean Absolute Percentage Error [Saxena et al., 2008a]:

$$\text{sMAPE} = \frac{100}{J} \sum_{j=1}^{J} \frac{2|\Delta_j|}{r_j + r_j^*}$$

2.5.2.2 Precision based metrics

Precision based metrics evaluate the spread of the prediction errors. Note that the precision metrics is invariant to the bias. In other words, an algorithm with large bias but low StdE and MAD can be adjusted to remove bias for improved accuracy.

- Standard deviation of the error [Saxena et al., 2008a]:

$$\text{StdE} = \sqrt{\frac{1}{J-1}\sum_{j=1}^{J}(\Delta_j - \text{Bias})^2}$$

- Mean absolute deviation from the sample median [Saxena et al., 2008a]:

$$\text{MAD} = \frac{1}{J}\sum_{j=1}^{J}|\Delta_j - \text{median}(\Delta_j)|$$

2.5.3 Performance evaluation framework for prognostics

RUL prediction in prognostics is made based on the history of condition measurements. Predictions made early on have access to less information about the dynamics of fault evolution and are required to predict farther in time, which makes the prediction task more difficult as compared to predicting at a later stage [Saxena et al., 2010]. The traditional performance metrics seldom take this factor into consideration and therefore is inadequate to represent the true performance of an algorithm. [Saxena et al., 2010] gave an attempt to address this problem by proposing a performance evaluation framework that consists of four performance metrics in hierarchy. Under this framework, the predictions made for one instance at different timestamps are treated as a whole; performance metrics are defined on the prediction series for each instance instead of single predictions. The four performance metrics under this framework are listed below.

Prediction Horizon evaluates how long time ahead of failure the algorithm can predict with at least the specified accuracy. The specified accuracy is given as a percentage of the actual life of the instance.

$\alpha - \lambda$ *performance* evaluates whether or not the predictions at a time within the prediction horizon stays within the accuracy requirement. The accuracy requirement is given as a percentage of the actual RUL (not the total life), which is a more strict requirement for accuracy.

Relative accuracy quantitatively evaluate the absolute percentage error of a prediction at a time within the prediction horizon, if the algorithm has met the requirements of the previous metrics.

Convergence evaluate how fast the prediction performance (any accuracy based metric) improves towards the end life of the instance, if the algorithm has met the requirements of the previous metrics.

This performance evaluation framework will be adopted in this thesis with modification in the case study (see Section 5.2).

2.6 Challenges of RUL prediction for complex systems

Although many algorithms have been developed and experimented for RUL prediction, successful stories on prognostics are still rare in reality. Multiple issues are challenging the application of prognostics in complex systems.

2.6.1 System complexity

For a complex system, it is almost impossible to understand and model every detail of the system's behavior. What's modeled is usually limited to the overall operating mechanism of the system (system model) and some micro-level physics models for the critical components in the system. There prognostics functionalities with more

coverage of the system's failure modes has to resort to the collected condition data.

However, a complex system may exhibit highly nonlinear, stochastic behavior due to various reasons, such as manufacturing variations of the vast number of components and their assembly, numerous fault and failure modes that may evolve or occur simultaneously, and the nonlinear relations between those factors and the measured signals. The nonlinear, stochastic behavior of the system has placed great challenge to data-driven modeling. In addition, the system behavior from different instances may behave inconsistently even under identical operational conditions, which adds a new level of challenge for the derived data models to make predictions.

2.6.2 Data quality

Prognostics application relies on the measured condition data of the system. Data quality issue further increases the challenge of prognostics design due to the fact of so-called "Garbage In, Garbage Out". Instrumentation of the system is seldom, if ever possible, perfect. Instrumentation accuracy can be improved by the advancement of new sensing technologies and data acquisition systems; however, the parameters to be measured can never be complete for a complex system. The system's operational settings as well as some environmental parameters may not have been measured completely; and the system's behavior may have been observed from only a limited number of perspectives (e.g. vibration measurement at a few locations in a huge machine). Without these measurements, the behavior variety of the system caused by them will have to be treated as noise during modeling.

Prognostics techniques model system degradation, which require the actual degradation to be given. However, degradation is an unobservable virtual property that can be only be defined vaguely based on the observables or the human judgment. When the data is collected from a complex system, the actual degradation towards

different faulty modes are unknown. Due to incomplete knowledge to the system, not all the fault reasons can be labeled by experience; actually some minor fault mechanism are never known and never measured. The many unknown information for the collected data create another challenge for data-driven modeling.

In many cases, the data used for modeling are collected from controlled experiments rather than from the actual system under its normal operation. Though these data may have higher "quality" in terms of noise level and completeness of context information about the system's actual condition, these data may not be representative of the actual system. Models developed from these data may over simplify the problem and produce poor performance once deployed.

2.6.3 Uncertain future operations

Operational conditions has great impact to the system behavior as well as the degradation progression pattern. Past operational conditions can be measured to facilitate prognostics modeling whereas future operational conditions are unknown. Therefore it has to be assumed that the system's future operation or usage will follow a certain estimated pattern, e.g. identical to the past. The uncertainty of future operation pattern adds a challenge for degradation estimation and RUL prediction.

In addition, a real-world system in the field are almost always maintained regularly with preventive maintenance practices. These maintenance actions will recover the system's condition and vary its behavior. Even if the time of preventive maintenance actions can be modeled (e.g. in semiconductor industry the equipment usually experiences major maintenance after a fixed number of hours of usage), the impact of them is hard to model. This creates a challenge for a deployed prognostics solution.

2.6.4 Validation difficulties

RUL prediction is different from future behavior predictions, such as weather or stock forecasting, which can be validated immediately after the future time comes to present and the true behavior becomes available. RUL prediction can only be validated after the system has experienced the whole life cycle and come to a real failure point; this process can take very long time (e.g. many years) for many engineering systems. If the system is actually approaching to the failure point, there's no reason to let the system fail without taking preventive actions. The actions will eventually change the original degradation process of the system and thus will invalidate the predictions made previously. In addition, certain run-to-failure experiments require frequent disassembly of the system to find out the ground true conditions; this assembly-disassembly process creates variations in the system performance and the system life shifts from what it may have been in the beginning of the experiment [Saxena et al., 2010].

Another challenge of validation comes from the lack of common performance metrics as mentioned in the previous section. An algorithm's strength and weakness are not absolute; they vary under different situations of the application.

3 Methodology of TSBP

In this chapter, the framework of Trajectory Similarity Based Prediction (TSBP) for RUL estimation will be laid out and the procedures of TSBP will be described in details.

3.1 Intuition

As mentioned before, the degradation process for many engineering systems, especially mechanical systems, is irreversible unless the condition is recovered by effective maintenance actions. The irreversible degradation process does not necessarily imply that the observed features will exhibit monotonic progression pattern during degradation. Such progression pattern is sometimes hard to model using parametric methods.

Considering a degradation process involving no or limited maintenance, the process may compose of a sequence of irreversible stages (either discrete or continuous) from new to worn out, which can be implicitly expressed by the trajectory of the measured condition data or features. Therefore the RUL of the system can be estimated if its future degradation trend can be projected from those historical instances that has failed. Inspired by the Instance-Based Learning (IBL) methodology, similarity between the degradation trajectories of different instances can be computed first; and then the failure time of one instance can be estimated based on the actually failure time of similar instances; finally the RULs estimated from multiple historical instances can be aggregated to generate the final prediction of RUL.

3.2 Problem definition

Let $\mathbf{x} = (x_1, x_2, \ldots, x_N)^T$ be an N-dimensional feature vector extracted from the raw data collected from a system. The feature extraction methods are application dependent and will not be discussed here. Suppose the system is instrumented at a certain time interval (variable or fixed) through its life cycle. The samples of the feature vector $\mathbf{x}_i$ is timestamped by t_i (or indexed by the cycle number i). Let t_E denote the end-of-life time stamp and t_I denote the time stamp of the latest measurement cycle. Then the RUL estimation problem for a system can be defined as computing RUL $r_I := t_E - t_I$ of a test instance given

1. The up-to-date history of feature vectors from the test instance $X_I = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_I\}$,
2. L training instances of the same type of system with complete history of feature vectors ${}^lX_I = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_I\}, l = 1, ..., L$, and
3. The known or pre-estimated end-of-life time ${}^lt_E$ for each training instance. If the training instance has failed at the last measurement cycle, then ${}^lt_E = {}^lt_I$; otherwise an RUL estimate ${}^l\text{RUL}$ for the training instance should be provided such that ${}^lt_E = {}^lt_I + {}^l\text{RUL}$.

In case the system operates under variable operating conditions, it is desirable to include measurements on the operating conditions as well for both the test instances and training instances.

3.3 Framework of TSBP

TSBP is a non-parametric prediction technique designed especially for engineering system RUL prediction problem. In TSBP, a number of degradation trajectories are extracted from the historical data of the training instances that have known failure

times, and form a library of degradation models. For a test instance of the same type of system, similarity between it and each of the models in the library are evaluated by computing the minimal weighted Euclidean distances between two degradation trajectories. Then the RUL of the test instance is estimated by the known failure time of each of the degradations models. The final RUL prediction is obtained by aggregating the multiple RUL estimates from individual degradation models.

As shown in Fig. 3.1, the TSBP method includes three essential procedures:

1. *Degradation trajectory abstraction*: build instance/local models from the degradation trajectories of the training instances.
2. *Similarity evaluation*: evaluate similarity between a test instance and each of the instance models based on the degradation trajectory; an RUL estimate will be obtained from each instance model.
3. *Model aggregation*: aggregate the RUL estimates obtained from all the instance models to get the final RUL prediction.

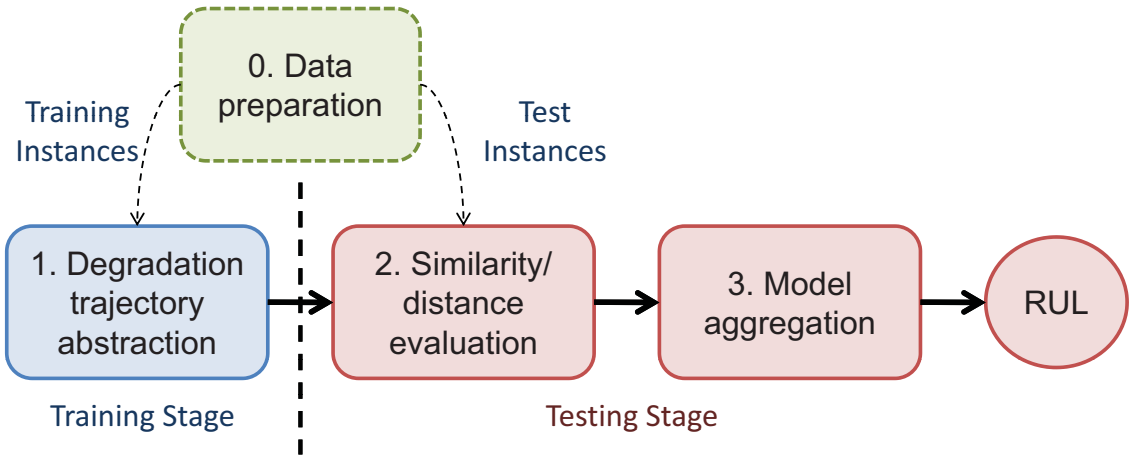

Figure 3.1: General procedures of TSBP approach for RUL estimation

In many cases, the Principal Component Analysis (PCA) will be applied first to reduce the dimensionality of the feature vector by converting it into a fewer number of principal components before taking procedure 1. PCA can remove linear correlation among the variables and suppress noise by fusing multiple variables. This process can be considered as an optional procedure 0, *Data preparation*, as shown in Fig. 3.1. Depending on the application, data preparation may include extra processing tasks, such as variable selection, data normalization, etc., which will be discussed in Chapter 4). This chapter will cover only the three essential procedures.

3.4 Key procedures

3.4.1 Degradation trajectory abstraction

Suppose $\mathbf{z} = (z_1, z_2, \ldots, z_M)^T$ consists of the first M Principal Components (PCs) (uncorrelated variables) computed from the N-dimensional ($N \geq M$) feature vectors $\mathbf{x}$, and $Z_I = \{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_I\}$ denotes the PC time series. Let ${}^lG$ denote the degradation model built from ${}^lZ_I$, the PC time series of the l^{th} training instance. The degradation model ${}^lG$ describes the PCs $\mathbf{z}$ as a function of time t:

$$ {}^lG : \mathbf{z} = {}^l\mathbf{g}(t) + \varepsilon, 0 \leq t \leq {}^lt_I \tag{3.1}$$

where ε is the noise term and in many cases is modeled as Gaussian.

Many parametric and non-parametric methods can serve the purpose to build such a degradation model. Many real-world systems can be assumed to follow a monotonic degradation pattern, but which may not be necessarily reflected on the measured data or features. For instance, RMS of a bearing's vibration, or its energy around a certain defect frequency, may not be monotonic increasing over the bearing's life cycle. Therefore sometimes it is hard to use a parametric model to model the progression of the observed features. One solution is to estimate the

actual, hidden degradation process of the system using, for instance, the Hidden Markov Models (see Section 2.4.4.5). Another solution is to employ non-parametric modeling techniques, such as Kernel Regression [Russell and Norvig, 2003], Locally Weighted Scatterplot Smoothing (LOWESS) [Craven and Wahba, 1979], Cubic Spline Interpolation [Cleveland, 1979], Relevance Vector Machines [Tipping, 2001], etc.

The selection of parametric or non-parametric methods can be flexible depending on the application. Considerations include but not are limited to the system's global degradation pattern, short-period characteristics, amount of available data, data noise level, and so on. As for engineering system RUL estimation, the focus is usually put on the long-term degradation behavior of the system; local fluctuations in the degradation trajectory can be treated as disturbance or noise. Therefore, in many cases a simple method consisting of a smoothing operation of the time series using a moving average filter followed by linear interpolation can serve the purpose.

In this thesis, four methods for degradation trajectory modeling are evaluated and compared. All methods discussed here model a 1-D variable as a function of time. If the time series of features or PCs have more than one dimension, multiple models have to be employed to model each of the dimension.

3.4.1.1 Exponential curve fitting

Exponential curve fitting is a non-linear regression problem with exponential models. This method was used to model a normalized feature trajectory during the 2008 PHM data challenge competition [Wang et al., 2008]).

The generic form of an exponential model for 1-D data is given as follows:

$$z = a \cdot \exp(b \cdot t + c) + d + \varepsilon \tag{3.2}$$

where a, b, c, d are four parameters to be learned from the training data. In a certain

case where a failure threshold z_f can be set for z based on knowledge (e.g. $z = 0$ when the features are normalized to [0 1] with 0 meaning the failure condition), the following constraint can be applied:

$$z_f = a \cdot \exp(b \cdot t_E + c) + d \tag{3.3}$$

where t_E is the failure time of the training instance. After solving the parameter d, the following form of exponential model can be applied:

$$z = a \cdot (\exp(b \cdot t + c) - \exp(b \cdot t_E + c)) + z_f + \varepsilon \tag{3.4}$$

The model parameters can be solved using non-linear least-square optimization methods. However, for an exponential model, it is not uncommon that the non-linear least-square solver diverges when the initial values of the parameters are poorly estimated. This issue can be significant when the data is very noisy or demonstrates little trend. It is also hard to come up a robust, automated method to estimate the initial values, which has made this method less desirable for automated computation.

3.4.1.2 Moving average filter and interpolation

A moving average filter provides the most simplest non-parametric solution to model a time series. With window size $2n+1$, a moving average filter takes the form

$$\underline{z}_i = \frac{1}{2n+1} \sum_{j=i-n}^{i+n} z_i \tag{3.5}$$

At the boundary of the time series, e.g. those samples at $i \leq n$ or $i \geq E - n + 1$, it is not recommended to use zero-padding because it tends to create large distortion. Instead, a reduced window size can be applied at the boundaries. Then the moving

average filter can be given as

$$\underline{z}_i = \begin{cases} \frac{1}{2i-1}\sum_{j=1}^{2i-1} z_i & \text{if } 1 \leq i < n+1 \\ \frac{1}{2n+1}\sum_{j=i-n}^{i+n} z_i & \text{if } n+1 \leq i \leq E-n \\ \frac{1}{2(E-i)+1}\sum_{j=2i-E}^{E} z_i & \text{if } E-n < i \leq E \end{cases} \tag{3.6}$$

For those points at $t \neq t_i, \forall i$, interpolation of neighboring points can be used to calculate $z(t)$. The simplest linear interpolation has the form

$$z(t) = \frac{(t_{k+1}-t)\underline{z}_k + (t-t_k)\underline{z}_{k+1}}{t_{k+1}-t_k}, t_k \leq t < t_{k+1} \tag{3.7}$$

Other higher order interpolation methods such as Cubic Spline Interpolation [Cleveland, 1979] can also be used.

3.4.1.3 Kernel regression smoothing

The kernel regression method has the following form

$$z(t) = \frac{\sum_{i=1}^{E} K(t,t_i) z_i}{\sum_{i=1}^{E} K(t,t_i)} \tag{3.8}$$

where $K(\bullet,\bullet)$ is the kernel function. In most cases, the Gaussian kernel are used:

$$K_G(x,y) = \exp\left(-\frac{\|x-y\|^2}{2\rho^2}\right) \tag{3.9}$$

where ρ is the kernel width, a free parameter that has to be specified based on the data, usually through cross-validation.

3.4.1.4 Relevance vector machines

RVM is a sparse kernel regression method that can be used for time series smoothing. As discussed in Section 2.4.4.6, RVM uses the relevance vectors, a much fewer number of samples from the training samples, to construct the regression model:

$$z(t) = \sum_{i \in \mathcal{S}} K(t,t_i) w_i \tag{3.10}$$

where $\mathcal{S}$ is the set of the indices of the relevance vectors z_i; w_i is the weights associated with the relevance vectors; $w_i = 0$ for $i \notin \mathcal{S}$. The number of elements in $\mathcal{S}$ is far less than E, the total number of samples in the time series Z_E. Again, the Gaussian kernel is the most frequent choice as the kernel function.

3.4.1.5 Comparison

Figure 3.2 shows the smoothing results provided by the four methods applied to a noisy feature series from a degradation process. The window size for (a) is 10 samples and the kernel width for (c) and (d) are both 10 samples. It shows that (c) produces a smoother curve than (b) and (d) with comparable kernel width and window size.

The characteristics of the four methods are summarized in Table 3.1. As a parametric method, the exponential curve fitting method has the most compact parameters once the model is trained and can achieve the fastest evaluation speed. However, the most critical drawback for this method is that the nonlinear least-square regression used to solve the model parameters may not converge when the time series does not exhibit good trends, or the initial estimates of parameters are inappropriately specified. Also the method assumes the data has exponential or at least monotonic trend; any minor fluctuation in the time series will be neglected by the fitted model.

The moving average filter and interpolation method provides a simple implementation of non-parametric smoothing. It is fast in both model training and evaluation. The kernel smoothing method can easily achieve a good smoothing performance; however it requires storing all the original data for model evaluation and its computational load can be very high during model evaluation if the number of original samples is large. RVM, as a sparse Bayesian method, greatly reduces the computational load compared with the kernel smoothing method; however it does it at the cost of increased training time. Also the model evaluation speed of RVM is

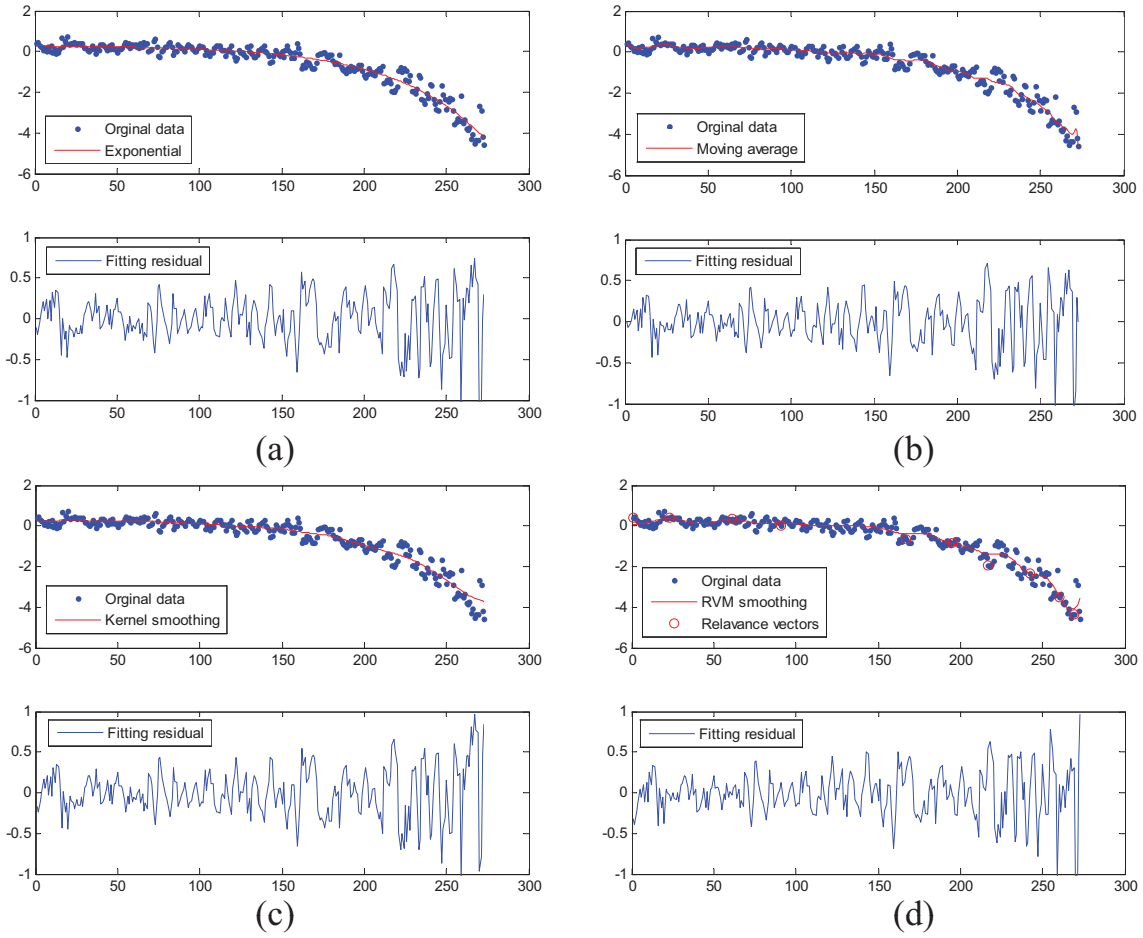

Figure 3.2: Smoothing noisy time series. (a) Exponential curve fitting (b)Moving average (c)Kernel smoothing (d) Relevance Vector Machine

not as fast as parametric methods or interpolation methods.

Therefore, we propose to use kernel smoothing method for training, but store a smoothed time series; during testing, interpolation to the smoothed samples will be used. The method requires the least computation in training and testing, and can still produce an acceptable level of accuracy in degradation trajectory abstraction.

[2] The RVM code used in this study comes from the open source Matlab toolbox "SparseBayes V2" (http://www.miketipping.com/index.php?page=rvm).

Table 3.1: Comparison of four data smoothing methods

Method	Exponential curve fitting	Moving average with interpolation	Kernel smoothing	RVM [2]
Smoothing power	Perfect	Low	High	Medium
Training speed	Fast	Fast	Not required	Slow
Evaluation speed	Fast	Fast	Slow	Medium
Storage	Low	High	High	Medium

3.4.2 Similarity evaluation

Definition of similarity/distance between instances plays an importance role for an IBL method. Most of the IBL applications reported define distance in an n-dimensional Euclidean space. For the problem of RUL estimation presented here, an instance is represented by a n-dimensional time series. The distance can be defined intuitively by the average Euclidean distance over multiple cycles between a test instance and a degradation model, i.e. $\frac{1}{I}\sum_{i=1}^{I}\|\mathbf{z}_i - \mathbf{g}(t_i)\|^2$. This definition is equivalent, to some extend, to the distance definition derived from the likelihood functions as shown below.

The M variables in $\mathbf{z}$ are linearly independent and with Gaussian variance σ_m^2 for each variable respectively. Then the likelihood that the measurement Z_I of a test instance comes from the degradation model ${}^lG$ is given as follows.

$$
\begin{aligned}
L(Z_I|{}^lG) &= \prod_{i=1}^{I}\prod_{m=1}^{M}\left((2\pi\sigma_m^2)^{-1/2}\exp\left(-\frac{\left(z_{mi}-{}^lg_m(t_i)\right)^2}{2\sigma_m^2}\right)\right) \\
&= \left(\prod_{m=1}^{M}(2\pi\sigma_m^2)^{-\frac{I}{2}}\right)\cdot\exp\left(-\sum_{i=1}^{I}\sum_{m=1}^{M}\frac{\left(z_{mi}-{}^lg_m(t_i)\right)^2}{2\sigma_m^2}\right) \\
&= \left(\prod_{m=1}^{M}(2\pi\sigma_m^2)^{-1/2}\cdot\exp\left(-\frac{1}{I}\sum_{i=1}^{I}\sum_{m=1}^{M}\frac{\left(z_{mi}-{}^lg_m(t_i)\right)^2}{2\sigma_m^2}\right)\right)^I
\end{aligned}
$$

The *Similarity* between Z_I and ${}^lG$ can be defined as the I^{th} root of the likelihood:

$$
{}^lS := \left(L(Z_I|{}^lG)\right)^{\frac{1}{I}} = \prod_{m=1}^{M}(2\pi\sigma_m^2)^{-1/2}\cdot\exp\left(-\frac{1}{I}\sum_{i=1}^{I}\sum_{m=1}^{M}\frac{\left(z_{mi}-{}^lg_m(t_i)\right)^2}{2\sigma_m^2}\right)
$$

Since the term $\prod_{m=1}^{M}(2\pi\sigma_m^2)^{-1/2}$ is constant for all, the definition of similarity can be further simplified as

$$
{}^lS := \exp\left(-\frac{1}{I}\sum_{i=1}^{I}\sum_{m=1}^{M}\frac{\left(z_{mi}-{}^lg_m(t_i)\right)^2}{2\sigma_m^2}\right) \tag{3.11}
$$

The squared distance is defined as negative log of similarity:

$$
{}^lD^2 := -\log{}^lS = \frac{1}{I}\sum_{i=1}^{I}\sum_{m=1}^{M}\frac{\left(z_{mi}-{}^lg_m(t_i)\right)^2}{2\sigma_m^2} \tag{3.12}
$$

This distance definition has the form of normalized Euclidean distance averaged over all cycles.

The use of the I^{th} root of the likelihood to define similarity ${}^lS$ has two purposes. Firstly, when using fewer samples from Z_I, one can still compute a valid distance, because the distance is defined as an average over the actual number of sum squares. It is common that a test instance, while still in good condition, already has a longer history than a training instance that has failed (i.e. $t_I > {}^lt_E$). Apparently some samples in Z_I will have to be dropped out because there are no definition for the corresponding ${}^lg(t_i)$. This situation also happens when a time lag between Z_I and ${}^lG$

is applied (see Section 3.4.2.1). Secondly, using I^{th} root of the likelihood can avoid underflow in numeric precisions while computing the distance.

An issue with this distance definition is that, it aligns the first cycle of Z_I to the first cycle of ${}^lG$, which may be a too strict definition for similarity, too strict to accommodate the minor discrepancies in the degradation process. Some modifications to the distance definition can be made to relax the evaluation of similarity. In this section, three improved distance definitions are proposed to describe instance similarity. Through these definitions, the similarity discrepancy between the instances can be accommodated from different engineering perspectives.

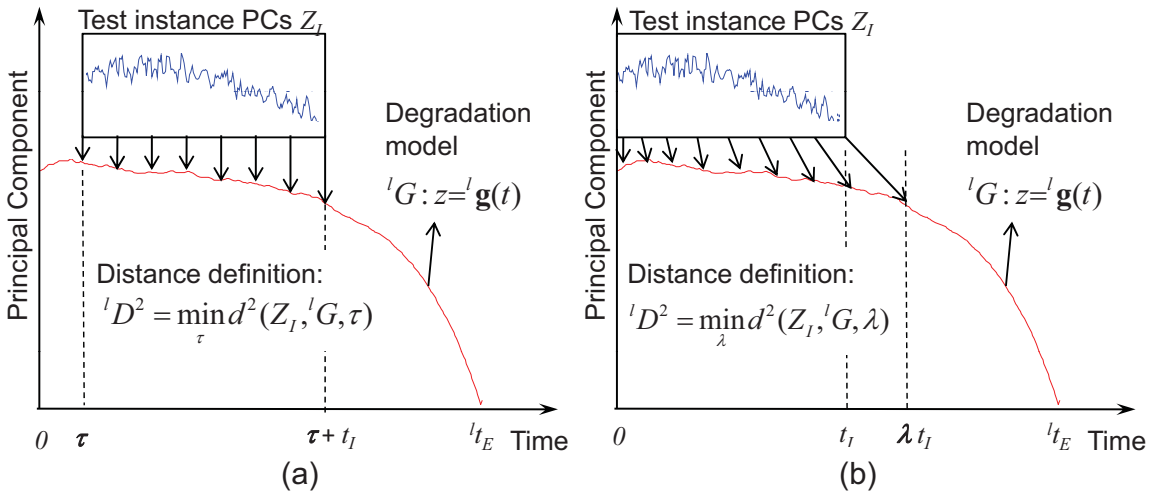

Figure 3.3: Distance definitions between a test instance and a degradation model. (a) MED-TL distance (b) MED-DA distance

3.4.2.1 Minimal Euclidean Distance with Time Lag (MED-TL)

In the real world, the test instance and the training instance may take a different length of initial stage to reach a similar degradation level. Therefore the definition of

similarity has to accommodate this difference in the initial conditions of two instances. To do this, one can compare the test instance against a certain time period in the training instance's history, searching for the best time lag τ that can minimize the Euclidean distance, as shown in Fig. 3.3(a). Therefore the squared distance between a test instance Z_I and an instance model ${}^lG$ can be defined as:

$$\begin{aligned} {}^lD^2 &= \min_{\tau} d^2(Z_I, {}^lG, \tau) \\ d^2(Z_I, {}^lG, \tau) &= \frac{1}{I} \sum_{i=1}^{I} \sum_{m=1}^{M} \frac{(z_{mi} - {}^lg_m(t_i + \tau))^2}{2\sigma_m^2} \end{aligned} \tag{3.13}$$

where τ is the time lag between the test instance trajectory Z_I and the model ${}^lG$, t_i is the time stamp of $\mathbf{z}_i$. At those $t = t_i$ where ${}^lg(t)$ is not defined (i.e. $t_i + \tau < 0$ or $t_i + \tau > {}^lt_I$), the corresponding term $(z_{mi} - {}^lg_m(t_i + \tau))^2/2\sigma_m^2$ will be dropped out, and distance averaging will be based on the number of the effective distance terms rather than the fixed number I.

A general constraint for τ applies: $0 < t_I + \tau < {}^lt_E$. In practice, a tighter constraint for τ can be specified, e.g. to confine ${}^lt_E - \tau$ by the range of the expected minimal and maximal life of the system.

For the distance definition given by Eq.(3.13), the distance terms for all the cycles have equal contribution to the total distance. In reality, however, it is reasonable to assume that the most recent cycles can explain more of the instance's degradation behavior than the earlier cycles do. Therefore, non-uniform weights with emphasis on the more recent cycles can be used. The squared distance can be expressed in a more general form:

$$\begin{aligned} {}^lD^2 &= \min_{\tau} d^2(Z_I, {}^lG, \tau) \\ d^2(Z_I, {}^lG, \tau) &= \frac{1}{\sum_{i=1}^{I} v_i} \sum_{i=1}^{I} \left(v_i \cdot \sum_{m=1}^{M} \frac{(z_{mi} - {}^lg_m(t_i + \tau))^2}{\sigma_m^2} \right) \end{aligned} \tag{3.14}$$

where v_i is the weights applied to different cycles of the test instance trajectory. For uniform weights $v_i = 1, \forall i$, this definition becomes Eq.(3.13). For non-uniform weights, the following weights based on Radial Basis Function can provide a desirable weight distribution:

$$\begin{aligned} v_i =& \exp\left(-(t_i - t_I)^2/2\rho^2\right) \\ \rho =& \gamma \cdot {}^l r_E \end{aligned} \tag{3.15}$$

where t_i is the time stamp of the i^{th} cycle of the the test instance, and ρ is the spread parameter that controls how fast the weights drop when a cycle gets far away from the most recent one at t_I. The choice of ρ is a percentage of the life ${}^l r_E$ of the degradation model ${}^l G$, controlled by the spread ratio γ (e.g. 0.5). The exact value of γ can be decided by heuristics, or optimized through cross-validation.

As mentioned before, the true life of the training instance will be used as a direct estimate to the test instance's life. Therefore the RUL estimate from the instance model can be easily computed:

$${}^l r_I = {}^l t_E - t_I - \arg\min_{\tau} d^2(Z_I, {}^l G, \tau) \tag{3.16}$$

3.4.2.2 Minimal Euclidean Distance with Degradation Acceleration (MED-DA)

Another method to relax the definition of similarity is to include a scaling factor λ in the distance definition to accommodate the degradation rate difference between the test instance and the model ${}^l G$:

$$\begin{aligned} {}^l D^2 :=& \min_{\lambda} d^2(Z_I, {}^l G, \lambda) \\ d^2(Z_I, {}^l G, \lambda) =& \frac{\max(\lambda, \frac{1}{\lambda})}{\sum_{i=1}^{I} v_i} \sum_{i=1}^{I} \left(v_i \cdot \sum_{m=1}^{M} \frac{(z_m i - {}^l g_m(\lambda \cdot t_i))^2}{\sigma_m^2} \right) \end{aligned} \tag{3.17}$$

where $\max(\lambda, 1/\lambda)$ is the penalty term for the difference in degradation rate. This distance definition can explain similarity between two trajectories when one of them has accelerated or decelerated degradation rate compared to the other, as shown in Fig. 3.3(b).

The scaling factor λ is naturally constrained by $0 < \lambda \cdot t_I \leq {}^{l}t_E$. Again, tighter constraints for λ can be applied, e.g. to confine ${}^{l}t_E/\lambda$ to the range of the expected minimal and maximal life of the system.

The RUL estimate produced by the model ${}^{l}G$ is given by:

$$ {}^{l}r_I = \frac{{}^{l}t_E}{\arg\min_\lambda d^2(Z_I, {}^{l}G, \lambda)} - t_I \tag{3.18} $$

3.4.2.3 Minimal Euclidean Distance with Time Lag and Degradation Acceleration (MED-TL-DA)

A combination of both the time lag and degradation acceleration factors leads to the following distance definition:

$$ \begin{aligned} {}^{l}D^2 &:= \min_{\tau,\lambda} d^2(Z_I, {}^{l}G, \tau, \lambda) \\ d^2(Z_I, {}^{l}G, \tau, \lambda) &= \frac{\max(\lambda, \frac{1}{\lambda})}{\sum_{i=1}^{I} v_i} \sum_{i=1}^{I} \left(v_i \cdot \sum_{m=1}^{M} \frac{(z_{mi} - {}^{l}g_m(\lambda \cdot t_i + \tau))^2}{\sigma_m^2} \right) \end{aligned} \tag{3.19} $$

The constraints for λ and τ include $0 < \lambda \cdot t_I + \tau \leq {}^{l}t_E$, as well as the estimated instance life $({}^{l}t_E - \tau)/\lambda$ to be within the expected minimal and maximal life of the system.

The RUL estimation produced by the model ${}^{l}G$ is then given as:

$$ \begin{aligned} {}^{l}r_I &= \frac{{}^{l}t_E - \tau^*}{\lambda^*} - t_I \\ (\tau^*, \lambda^*) &= \arg\min_{\tau,\lambda} d^2(Z_I, {}^{l}G, \tau, \lambda) \end{aligned} \tag{3.20} $$

3.4.2.4 Solving the optimization problem

As we can see that the three distance definitions each involve a bound-constrained non-linear optimization process to decide the time lag or degradation acceleration rate. The problem cannot be formulated as an non-linear least square problem (due to the varying number of summation terms) and no gradient function can be provided explicitly. To solve this problem, we choose to use the the Pattern Search algorithm [Torczon, 1997], which is a derivative-free global minimization algorithm implementing the direct search (coordinate search) method, and has been proved to have global convergence with general constraints and simple bounds [Lewis and Torczon, 2002].

3.4.3 Model aggregation

According to the definition of squared distance given by Eq.(3.12), the similarity score of the test instance given by each degradation model ${}^{l}G$ can be obtained

$$ {}^{l}S = \exp(-{}^{l}D^2) \tag{3.21} $$

All RUL estimates and the corresponding similarity scores form a set $\mathcal{H}_I = \{({}^{l}r_I, {}^{l}S_I) | l = 1, 2, ..., L\}$. The goal of model aggregation is to aggregate the multiple estimates in $\mathcal{H}_I$ to get the final prediction of RUL.

The simplest method of aggregation is to use similarity-weighted sum, which provides a *Point Estimate* of the RUL:

$$ r_I = \frac{\sum_{l=1}^{L} {}^{l}S \cdot {}^{l}r_I}{\sum_{l=1}^{L} {}^{l}S} \tag{3.22} $$

In most real world applications, point estimation of RUL is inadequate for uncertainty management of predictions and thus hard to be utilized for decision making. A probability distribution, or a confidence interval for the predicted RUL is very

important. Therefore, density estimation methods will be applied here to estimate the probability distribution of RUL from set $\mathcal{H}_I$.

As mentioned before, the TSBP method has been developed to address the problem of unknown failure modes mixed in the collected data. It is not surprising if the estimated RULs from the library of degradation models have a distribution of multiple modes. Therefor it will be hard to assume a certain parametric form of distribution for the set $\mathcal{H}_I$. Here, a non-parametric method called Kernel Density Estimation (KDE, a.k.a Parzen window method) [Sheather and Jones, 1991] is employed.

For n independent and identically-distributed samples $\{x_1, x_2, ..., x_n\}$ of a random variable x, the kernel density approximation of its probability density function is given by

$$\hat{f}_h(x) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h}K(\frac{x - x_i}{h}) \tag{3.23}$$

where K is the kernel function and h is the bandwidth. The frequent used Gaussian kernel is given as

$$K(\frac{x - x_i}{h}) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{(x - x_i)^2}{2h^2}\right) \tag{3.24}$$

In our application, each sample ${}^{l}r_I$ has a weight ${}^{l}S_I$, therefore the the density estimation of RUL will have the following form

$$\hat{f}_h(r_I) = \frac{1}{\sum_{l=1}^{L}{}^{l}S_I}\sum_{l=1}^{L}\frac{{}^{l}S_I}{\sqrt{2\pi}h}\exp\left(-\frac{(r_I - {}^{l}r_I)^2}{2h^2}\right) \tag{3.25}$$

The point estimation of RUL takes the median of r_I:

$$\hat{r}_I = \arg\min_{x} \quad \int_{-\infty}^{x}\hat{f}_h(x)dx \geq 0.5 \tag{3.26}$$

The bandwidth h is a free parameter that influence the resulting estimate and has to be specified or estimated from the data. As shown in Fig. 3.4, a larger

bandwidth will produce smoother estimation, but it might possibly distorts the original distribution.

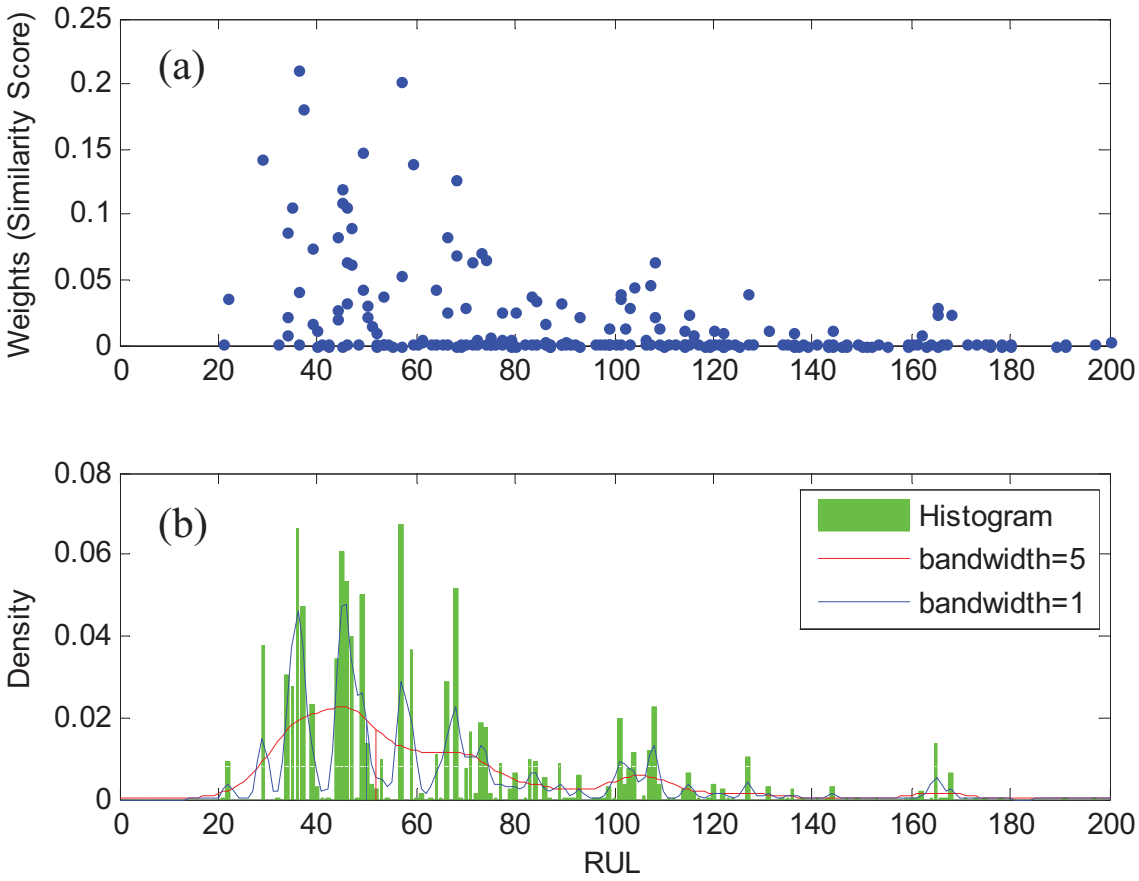

Figure 3.4: Kernel density estimation from samples with weights. (a) Samples and their weights (b) Weighted histogram and kernel density estimation with different bandwidth

Many data-driven bandwidth selection method has been proposed in literature and many have been implemented in commercial software packages. In this thesis, the KDE method via diffusion proposed by Botev et al. [2010] is used. It is reliable and extremely fast kernel density estimator for one-dimensional data with automatic

bandwidth selection. The advantage of the method is that, unlike many others, it does not assume a parametric model for the data and thus the estimation does not deteriorate for multi-modal densities with widely separated modes [Botev et al., 2010].

3.5 Assumptions of TSBP

In Section 2.3, the discussion on the scope of RUL modeling brought forward a few general assumptions on the feasibility of predictive analysis of RUL, as summarized below:

1. The system under study shows an irreversible evolving degradation behavior (gradually developing wear, faults, or anomalies) before a failure;
2. Catastrophic failures and infant failures are considered as exceptions for the period that RUL predictions are made;
3. The effect of the maintenance actions, if any, during the life cycle under consideration is negligible; otherwise, a life cycle is considered to be complete when a major corrective maintenance action is carried out.

For the TSBP approach, additional assumptions are made:

4. The system is instrumented at discrete time (called measurement cycles). During each measurement cycle, the system's operation causes an equivalent amount of wear.
5. The system's future operation follows a similar pattern as the past.

Note that for the training instances, run-to-failure data is desirable but not mandatory. Data collection does not necessarily start from brand-new of the system nor end with a failure cycle, as long as an estimate of the actual failure time of each training instances is provided.

4 Data Preparation for TSBP Modeling

In this chapter, the issues and techniques to convert the data to the desirable form for TSBP modeling will be discussed. The issue on data handling for variable operational conditions will be emphasized. At the end, issues beyond data preparation, i.e. data collection, will be discussed.

4.1 Overview

As discussed in Chapter 3, the framework of TSBP approach for RUL estimation include three essential procedures and one optional procedure named as *Data Preparation* (Fig. 3.1). The objective of data preparation is to convert the raw data into the desirable form for TSBP modeling. The transformed data should be a time series of M ($M \geq 1$) linearly independent variables that exhibits noticeable trend over the time. The time series is noted as $Z_I = \{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_I\}$ with time stamps $\{t_1, t_2, ..., t_I\}$, where $\mathbf{z}_i = (z_{1i}, z_{2i}, \ldots, z_{Mi})^T$.

The scope of data preparation include, but not limited to, the following aspects:

- *Feature extraction*: Extract meaningful information buried in the raw data.
- *Denoising*: Reduce noise of the data and deal with outliers. Denoising is usually applied to the raw data before feature extraction.
- *Data normalization*: Transform the data into a nominal range in case that the raw data are collected under different operational conditions so that the extracted features are not directly comparable across regimes.
- *Dimension reduction*: Either select a variable subset or transform the variables into a lower dimension.
- *Health assessment*: for certain prognostics models, it may be necessary to

compute a health index from the multi-dimensional features. The health index can be physics-related or purely mathematics-based. For the later case, the health assessment process can be treated as a special dimension reduction (with data normalization effect sometimes) that ultimately transform the data into a single dimension (e.g. a value between 0 and 1).

Denoising and feature extraction for a prognostics application are application dependent and will not be discussed further here. The other three aspects for data preparation, namely data normalization, dimension reduction and health assessment, will be discussed in this chapter to address a frequently encountered issue in prognostics applications: *data collected under variable operational conditions*.

4.2 Data handling for variable operating conditions

Operational conditions are a set of variables that decide the settings of the system's operation. They can be considered as "inputs" to the system in general, no matter they are explicit inputs such as control settings, or implicit settings such as environmental parameters, usage patterns, etc. For instance, the speed and feed rate for a machining process are operational conditions while the power consumption of the spindle is not; the speed, altitude and ambient temperature for an aircraft engine are operational conditions, while the its compressor outlet temperature is not.

Sometimes the operational conditions have to be obtained through measurement, such as the rotational speed of a rotary machine. These measurements can be treated as "conditions", if the difference between the measurement and the control setting is negligible for modeling purposes; otherwise they have to be treated as "outputs" or "behaviors" of the system. For instance, the speed measurement of an asynchronous induction machine can be used as the operational conditions of the motor in some cases of motor health assessment; however, it will be treated as an output variable

if the speed setting of the motor is also known from the controller, considering that there's synchronous speed slip for an induction machine and thus its actual speed is always less than the speed setting.

In many applications, the operational conditions have huge influence to the readings of "behavioral" measurements or the extracted features from the system, such that the feature time series will show large variance which overwhelms the trend caused by system degradation over time – this will create great challenges to degradation tracking for RUL modeling. Therefore, the data collected under variable operational conditions have to be preprocessed before applying TSBP modeling.

In this thesis, we consider the cases that the operational conditions of a system is explicitly available, either through control settings or measurements. Let $\mathbf{u}_i$ denotes the vector of variables for the system's operating conditions at time t_i. Together with the extracted features $\mathbf{x}_i$, the available data at each measurement cycle will be noted as a 3-turple $(t_i, \mathbf{u}_i, \mathbf{x}_i)$. If the system dynamics can be modeled physically or mathematically by function f with a set of parameters θ, $\mathbf{x} = f(t, \mathbf{u}; \Theta)$, then many model-based FDI approaches (see Section 2.4.3.2) can be employed to handle this issue. Here we consider only the cases that data-driven approaches are inevitable due to lack of system models.

The objective of data-driven preprocessing can be noted as the transformation

$$(t_i, \mathbf{u}_i, \mathbf{x}_i) \rightarrow (t_i, \mathbf{z}_i)$$

where the vector $\mathbf{z_i}$ is independent from $\mathbf{u}_i$.

One method to implement this transformation is to design a supervised, global black-box health assessment model, such as a Neural Network (NN), with both the operational condition $\mathbf{u}$ and the features $\mathbf{x}$ as input and with a virtual (no physical

meaning) health index as output, as shown in Fig. 4.1. The training data to learn the NN model can be prepared in the same way as described in the later section 4.4.2.

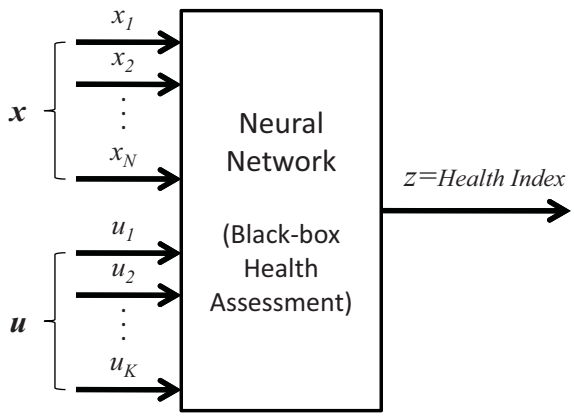

Figure 4.1: Data preparation for RUL modeling using Neural Networks

The global health assessment model, however, is hard to converge for a complex data set, and has high potential to over-fit the training data. This has led to the use of multi-regime approaches.

The multi-regime approaches are inspired by the Divide-and-Conquer philosophy, which decompose the problem through the use of local models in different operating regimes in the operation space of the system. Within each regime, the sensor measurements will have similar baseline and thus can be used to track the trend of degradation using local models. However, at each measurement cycle, the data may be collected from only one of the regimes; each regime may contain only a limited number of samples scattered at unevenly distributed time stamps. If each regime is considered independently, RUL modeling will have to face the challenge of insufficient samples. Therefore, it is desirable to first preprocess the data in individual regimes and then merge them together to form a new time series with the original time stamp.

After that, RUL modeling can be carried out on the new time series. To achieve this, two types of methods can be envisioned: one with health assessment in each operating regime to convert multivariate data into a 1-D health index, and another one with data normalization in each operating regime to normalize the multivariate data to a unified scale. The first method will generate a 1-D time series of health indices and lead to health-level RUL modeling; the second method will retain the original dimension of the features and lead to feature-level RUL modeling. Both method will be discussed in the subsequent sections.

4.3 Regime partitioning

Suppose the operational conditions $\mathbf{u}$ can be clustered into a finite number of operating regimes $\Omega = \{O_1, O_2, ..., O_P\}$ using a certain clustering or space partitioning algorithm $f_c(\mathbf{u})$. The output of f_c are fuzzy-logic-style membership scores to each of the clusters:

$$\mathbf{C} = f_c(\mathbf{u}) = (C_1, C_2, ..., C_P) \tag{4.1}$$

where $C_p = \text{Member}(\mathbf{u} \in O_p)$. The exact clustering algorithm can be arbitrary, such as k-means, Gaussian Mixture Models [Mclachlan and Peel, 2000], Fuzzy c-means [Sugeno and Yasukawa, 1993], etc., as long as the algorithm's output can be converted to the desired form as given by Eq.(4.1).

The method described above is the general regime partitioning approach with soft regime boundaries for continuous condition variables. In case of discrete operating regimes or regimes with hard boundaries, the output of the clustering algorithms can be simplified as the cluster index:

$$f_c(\mathbf{u}) = \arg \max_{k=1,..,P} C_k \tag{4.2}$$

Note that the regimes identified by the partitioning algorithms may not cover the full operation space of the system due to limited samples collected. It is possible for a sample $\mathbf{u}$ to have very low membership score, if not zero, to all the clusters. In practice a certain rule/threshold should be established to decide whether a sample should be clustered to none of the known regimes. If so, this special sample will not be passed to the local models for health assessment; the resultant time series will have a "missing sample" for this time stamp. If a certain on-line learning scheme can be developed for operating regime identification, this special sample may be labeled and saved for learning a new operating regime later when enough samples of such accumulate.

4.4 Multi-regime health assessment

Health assessment in nature performs two operations in one step, i.e. multivariate data fusion and normalization. Health assessment will transform the multi-dimensional features into a 1-D (virtual) health index within a normalized range. Therefore, the health indices obtained from each operating regime will become comparable and can be merged with the original timestamps to form a single 1-D time series, as shown in Fig. 4.2.

4.4.1 Generic multi-regime health assessment model

For each operating regime O_p, one local health assessment model with parameters $\Theta^{(p)}$ will be created:

$$z = h_{HA}^{(p)}(\mathbf{x}; \Theta^{(p)}) \tag{4.3}$$

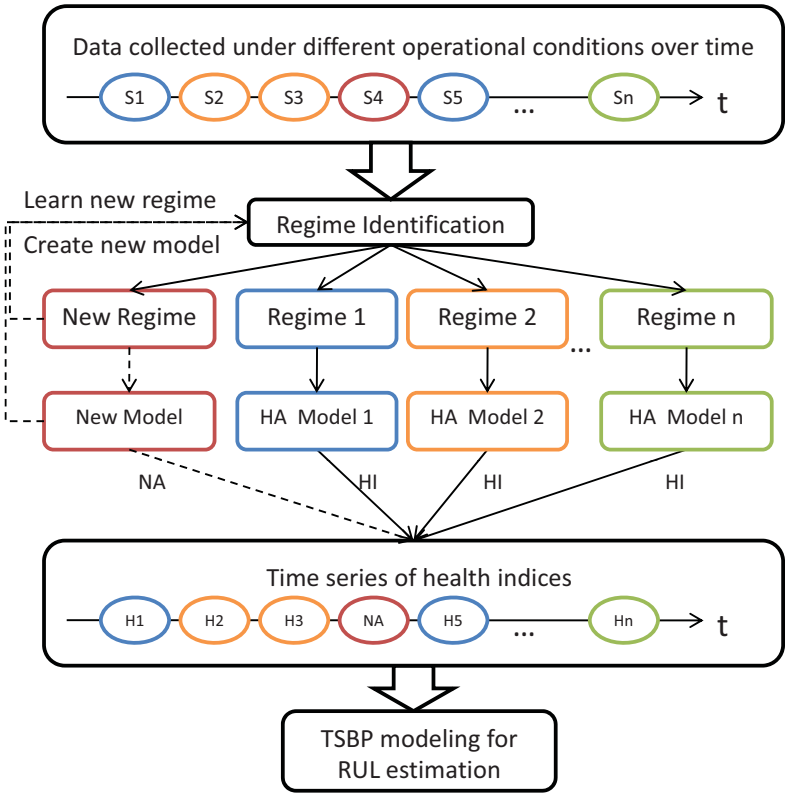

Figure 4.2: Data preparation for RUL modeling through multi-regime health assessment

For samples $(\mathbf{u}_i, \mathbf{x}_i)$, the following transformation can be applied to convert the samples into the health index domain:

$$z_i = \frac{\sum_{p=1}^{P} C_p \cdot h_{HA}^{(p)}(\mathbf{x}_i; \Theta^{(p)})}{\sum_{p=1}^{P} C_p} \tag{4.4}$$

$$C_p = \text{Member}(\mathbf{u}_i \in O_p)$$

In case of discrete regimes, health assessment can be simplified:

$$z_i = h_{HA}^{(p)}(\mathbf{x}_i; \Theta^{(p)}) \tag{4.5}$$

$$p = \arg \max_{k=1,..,P} C_k$$

If the health index is defined by a certain physical quantity computed from a certain physics-based local models, e.g. the crack length in a gear, its value will be comparable when obtained from different operating regimes. However, if the health index is computed from a data-driven method, one has to make sure the method produces a normalized health index.

Usually an unsupervised data-driven health assessment method that builds models solely on the baseline data (from normal condition of the system) will not qualify. For example, the methods using non-normalized distances to the baseline condition, such as Euclidean distance, Mahalanobis distance or Minimal Quantization Error (MQE) given by the Self-Organizing Maps [Huang et al., 2007; Kohonen, 2002], do not give a health index comparable under different operating regimes; the obtained distances have to be normalized to a range, usually between 0 and 1, using both the baseline and faulty data – this essentially becomes a supervised health assessment method.

4.4.2 Generic training method for a supervised health assessment model

The training data to learn a supervised health assessment model can be prepared with the following method:

- For those samples $(\mathbf{u}_i, \mathbf{x}_i)$ collected at early age of the system, e.g. $t_i < T_1$, set the corresponding output $z_i = 1$;
- For those samples $(\mathbf{u}_i, \mathbf{x}_i)$ collected at late age of the system, e.g. $t_i > T_2$, set the corresponding output $z_i = 0$;
- Those samples $(\mathbf{u}_i, \mathbf{x}_i)$ collected at middle age of the system, e.g. $T_1 \leq t_i \leq T_2$, will not participate model training.

The definition of early/late age of the system is controlled by parameters T_1 and T_2. A rule-of-thumb practice to choose the parameters is to use a certain percentage of the total life of the instance, e.g. $T_1 = t_E * 5\%$ and $T_2 = t_E * 95\%$. The obtained virtual

health indices z_i form a 1-D time series, which is ready for TSBP modeling. Once the training samples are prepared, they will be clustered based on the operational conditions $\mathbf{u}_i$ and used to train different health assessment models under different operating regimes.

4.4.3 Linear models for multi-regime health assessment

Many supervised health assessment model can be employed here. Yan et al. [2004] used a Logistic Regression model to convert the multi-dimensional features into health indices, which were then used to build an ARMA model to predict machine performance. The logistic regression model for health assessment has the following form

$$z = h_{HA}^{(p)}(\mathbf{x}; \Theta^{(p)}) = \frac{\exp\left(\theta_0^{(p)} + \sum_{n=1}^{N} \theta_n^{(p)} \cdot x_n\right)}{1 - \exp\left(\theta_0^{(p)} + \sum_{n=1}^{N} \theta_n^{(p)} \cdot x_n\right)} + \varepsilon \tag{4.6}$$

where $\Theta^{(p)} = (\theta_0^{(p)}, \theta_1^{(p)}, ..., \theta_N^{(p)})$ are N+1 model parameters for the p^{th} operating regime, and ε is the noise term. The method to prepare data for training the logistic regression model is similar to the generic training method described in the previous section.

The merit of a logistic regression model is that it will produce a health index within the range of 0 and 1 strictly. However, through the experiments performed in this thesis it is found that logistic regression, by its nonlinear nature, will distort the original degradation trend exhibited in the feature series; more specifically, the generated health index series becomes insensitive near the end life of the system which will lead to larger RUL prediction error using TSBP approach.

To preserve the original trend in the feature series, a linear regression model was

used for health assessment in the earlier research on TSBP [Wang et al., 2008].

$$z = h_{HA}^{(p)}(\mathbf{x}; \Theta^{(p)}) = \theta_0^{(p)} + \sum_{n=1}^{N} \theta_n^{(p)} \cdot x_n + \varepsilon \tag{4.7}$$

Unlike logistic regression, linear regression cannot ensure the health indices obtained to be strictly confined to the range of 0 to 1; however, this does no harm to RUL prediction using the TSBP approach, which does not rely on explicit thresholds of the health index to determine a failure condition.

4.4.4 Issues of linear models for health assessment

Both approaches discussed above belongs to (generalized) linear models. These models (a single model by itself) cannot fit well for data with mixture degradation trends. In other words, they require each variable to have consistent trend, either rising or falling, for all the training instances. For many real-world problems, however, there may be multiple failure modes, causing a certain feature to show different trends towards the end life in different instances. And in some cases, training instances with different failure modes may not have been classified adequately and thus have been mixed together in the training data, which will cause poor fitting using a (generalized) linear model.

Considering a case that run-to-failure data are collected from multiple instances of a system. Two features are extracted from the data. Feature 1 exhibits a rising trend over time when the system operates towards its end life. The same trend is shown for all the instances. When this feature from all the instances is plotted together in one figure with the last time stamp of each instance aligned, clear trend can be observed, as shown in Fig. 4.3. Feature 2, however, exhibits a rising trend when an instance fails due to one failure mode whereas exhibits a totally different trend for another failure mode. Therefore feature 2 at near-end-life of the system shows a large

variation. If feature 2 is included to build the local health assessment model, the model sensitivity will be compromised.

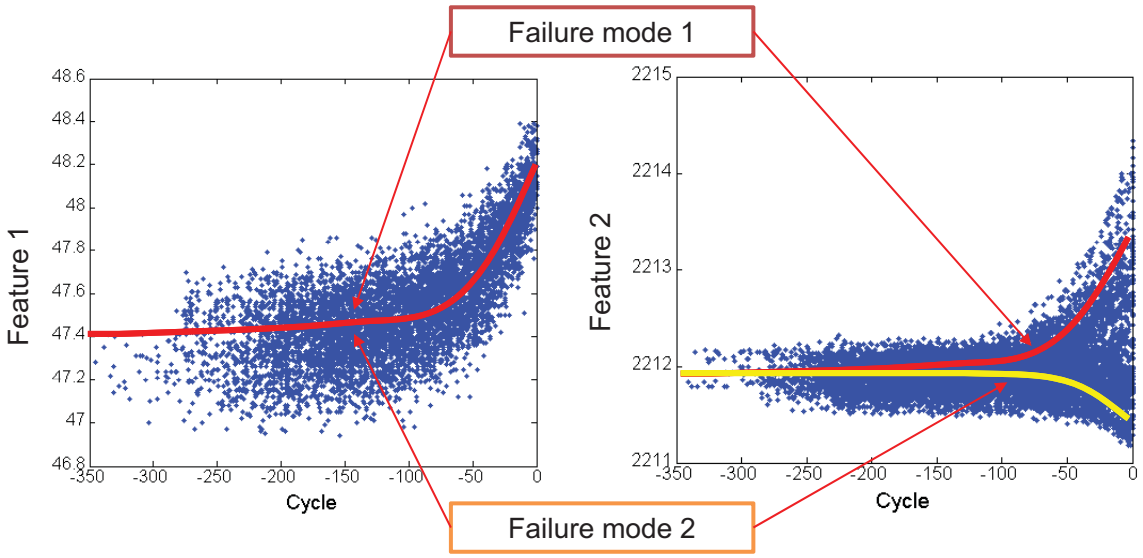

Figure 4.3: Features exhibit different trends when multiple failure modes exist

Therefore using generalized linear models will require careful selection of features to include in the model. Only those features that exhibit consistent trend for all the instances (failure cases) should be included. Obviously, excluding many other features for modeling will cause potential loss of certain important information regarding to the system's degradation and failure patterns.

4.4.5 Nonlinear health assessment models

One method to overcome the aforementioned challenge is to use a nonlinear model (or a combination of multiple linear models) for supervised health assessment. For example, a Neural Network can be used to map the features to the health index, $z = f_{NN}(\mathbf{x}; \Theta)$. Training of the NN model follows the generic training method for

supervised health assessment models (Section 4.4.2). Multiple neural networks have to be created for different operating regimes.

Another method to address the challenge is to use the MQE-based health assessment method [Huang et al., 2007] derived from the SOM. The faulty data (with multiple fault modes) can be used to train an SOM; for any feature vectors, the MQE can be computed to represent the level of "fault" – MQE in nature is the minimal distance to all the faulty samples. With a certain normalization process to the obtained MQE, a valid health assessment model(i.e. comparable health indices under different regimes) can be derived.

Nevertheless, the nonlinear health assessment methods will distort the original trend of the feature series somehow. In addition, by converting data with potentially multiple faulty modes into a 1-D health index, the difference in the degradation patterns will be neglected. This will eventually compromise the power of trajectory modeling using TSBP approach. Therefore in practice, the health-level RUL prediction is less preferable than the feature-level health assessment.

4.5 Multi-regime data normalization

The multi-regime data normalization approach is used to prepare data for feature-level RUL prediction. With this approach, the N features will be normalized individually into a nominal range under each operating regime and then form a new time series with the original feature dimension (Fig. 4.4). After that, the dimension of the new time series will be reduced through Principal Component Analysis (PCA) to remove linear correlations among the transformed features. The resultant data might still be multi-dimensional, but are ready for TSBP modeling. The key procedures of this approach will be discussed in details.

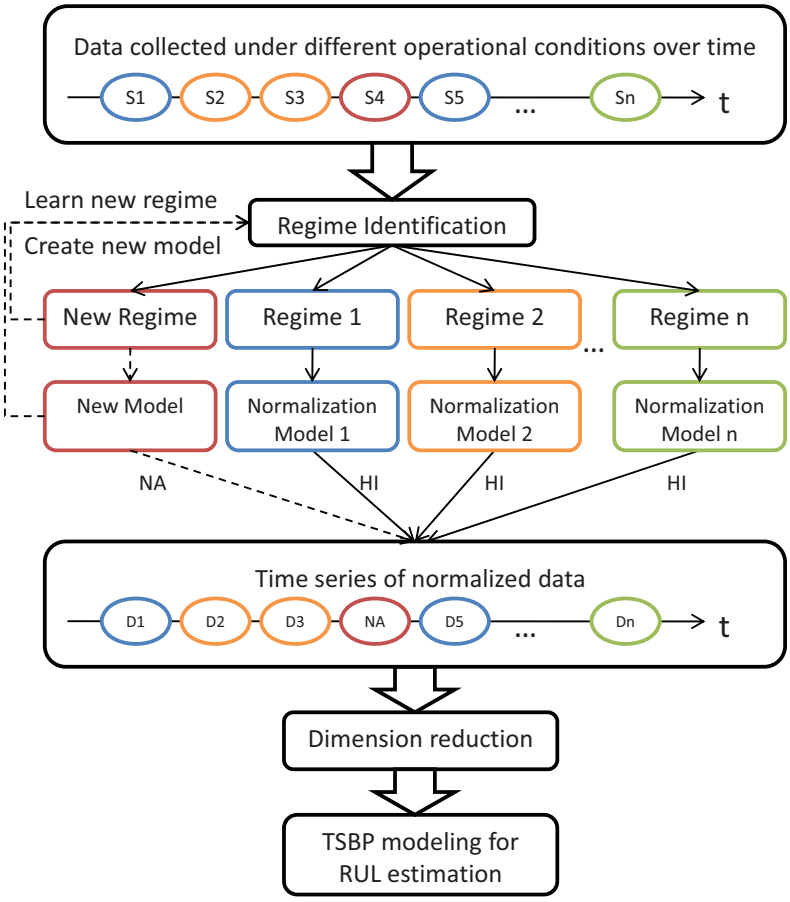

Figure 4.4: Data preparation for RUL modeling through multi-regime data normalization

4.5.1 Data normalization

Suppose each sample of $\mathbf{x}$ is classified into one of the P operating regimes O_p and forms sample set $\{\mathbf{x}\}^{(p)} = \{\mathbf{x}_i | \mathbf{u_i} \in O_p\}$ based on the clusters with hard boundaries determined by the operational conditions $\mathbf{u}$. Then the sample mean and standard

deviation for each regime can be computed

$$\bar{\mathbf{x}}^{(p)} = \text{Mean}(\{\mathbf{x}\}^{(p)}) \tag{4.8}$$

$$\mathbf{s}^{(p)} = \text{Std}(\{\mathbf{x}\}^{(p)}) \tag{4.9}$$

For the i^{th} cycle with regime membership $C_p = \text{Member}(\mathbf{u}_i \in O_p)$, the sample $\mathbf{x}_i$ will be transformed into $\mathbf{y}_i$ as follows:

$$\begin{aligned} \mathbf{y}_i &= (y_{1i}, y_{2i}, \cdots, y_{Ni})^T \\ y_{ni} &= \frac{\sum_{p=1}^{P} C_p \cdot (x_{ni} - \bar{x}_n^{(p)})/s_n^{(p)}}{\sum_{p=1}^{P} C_p}, n = 1, ..., N \end{aligned} \tag{4.10}$$

where x_{ni}, $\bar{x}_n^{(p)}$ and $s_n^{(p)}$ are the n^{th} element of the vectors $\mathbf{x}_i$, $\bar{\mathbf{x}}^{(p)}$ and $\mathbf{s}^{(p)}$. In case of discrete regimes, the transform can be simplified as

$$\begin{aligned} y_{ni} &= (x_{ni} - \bar{x}_n^{(p)})/s_n^{(p)}, n = 1, ..., N \\ p &= \arg \max_{k=1,..,P} C_k \end{aligned} \tag{4.11}$$

In order for the normalized features in each operating regimes to be equivalent, this normalization method assumes that an operating condition has identical probability to occur at any cycle:

$$Pr(\mathbf{u}_i \in O_p) = Pr(\mathbf{u}_j \in O_p), \forall i, j = 1, ..., E \tag{4.12}$$

This ensures the sample mean and standard deviation computed from those samples under an operating regime are unbiased estimation of the mean and standard deviation for the feature in the said regime. Finally, this will allow to merge the normalized feature in their original timestamps without altering the characteristics of the system's full degradation process. The resultant time series is noted as $Y_I = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_I\}$.

4.5.2 Variable selection vs. variable weighting

Variable selection (a.k.a. feature selection, sensor selection, variable subset selection) in data modeling has been an important topic for pattern analysis and machine learning. Inclusion of irrelevant or redundant variables may be detrimental to the performance of a prediction model, causing over-fitting or reduced sensitivity. The intuition of variable selection is to maximize the relevancy and minimize the redundancy in the variable subset.

There are two major variable selection approaches, the wrapper approach and the filter approach [Kohavi and John, 1997]. With the wrapper approach, performance of the target model (e.g. the classification rate for a classification model, or MSE for a regression model) built with different variable subsets is evaluated, and the optimal variable subset is the one that can maximize the model performance. The wrapper approach usually requires a cross-validation process to evaluate the model performance repetitively. If the number of variable candidates is small, it may be possible to performance an exhaustive search on all the combination of the variables, resulting 2^n possible subsets for n variables. When n is large, exhaustive search becomes infeasible. Many methods such as forward selection and backward elimination [Guyon and Elisseeff, 2003] have been designed to address the searching problem. Although the wrapper approach is straight forward and universal, it may require massive amount of computational, especially in the case that variable selection constitutes only a very preliminary task in a complex data modeling task. For the RUL prediction problem considered here, the TSBP approach involves multiple processing procedures. There are too many parameters that can affect the final performance. The optimal variable subset obtained using the wrapper approach may cause over-fitting to the problem.

The filter approach instead uses local performance criteria for variable subset selection, which is not tuned/bound to a specific target model. The filter approach may use different local criteria for variable selection, such as the correlation criteria, mutual information criteria, etc. [Guyon and Elisseeff, 2003]. It is also possible to define a simpler, local, linear predictor, which is different from the more complex global model, and whose performance can be used as a criteria for applying the wrapper approach (e.g. in [Bi et al., 2003]). Selection of local criteria is critical for the filter approach. For the RUL prediction problem, the criteria of variable relevancy can be defined on the ability to reveal the degradation trend of the system. One criteria can be defined to select those variables with the largest separation of the early-stage and near-end-life feature sets for the system. However, RUL modeling for a prognostics application is a regression problem; variable relevance in this case is more than the classification capability. For degradation modeling, intuitively, the slower characteristics in a feature time series reflect the system's degradation behavior whereas the fast characteristics are usually either noise or irrelevant to degradation. Here, the author proposed an empirical criteria, called Empirical Signal-to-Noise Ratio (eSNR), for variable relevance evaluation.

Let $\{s_i\}$ be a 1-D time series representing the features of the system evolving from early age to the end life. Let $\{\underline{s}_i\}$ be the time series after smoothing $\{s_i\}$ by a certain filtering or smoothing algorithm. Then the eSNR of this variable can be calculated as the ratio between the variance of the smoothed time series and the original time series:

$$\text{eSNR}(s) = \text{var}(\{\underline{s}_i\})/\text{var}(\{s_i\} \tag{4.13}$$

For a variable whose global variance (over the whole time series) is mainly attributed to the local variance (variance within a shorter period in the time series), the smoothed

time series will have a much smaller variance compared to the original, and thus will have a smaller eSNR value. Therefore variable selection can be performed based on the ranking given by eSNR.

However, variable selection based on single variable ranking has multiple limitations. It has been shown that a variable that is useless by itself can be useful when taken with others, and that two variables that are useless by themselves can be useful together [Guyon and Elisseeff, 2003]. Another problem is that it is hard to decide the number of high-ranking variables to retain. Intuitively, fusion of more relevant variables may help to suppress noise and reduce uncertainty exhibited by individual variables. In addition, Hall [1999] argued that feature selection may degrade model performance in cases where some variables are eliminated which are highly predictive of very small areas of the instance space.

These reasons have led us to reconsider the problem of relevancy and redundancy of variables from another perspective, i.e. the variable weighting approach Hall [1999]. The variable weighting approach can preserve variable relevancy; however, it does not reduce the number of variables and thus does not reduce variable redundancy. Therefore, the Principal Component Analysis (PCA) (see Section 4.5.3) can be applied following variable weighting to remove correlation or redundancy.

In this thesis, the variables are weighted based on their eSNR given by Eq.(4.13). Each variable y_n in the normalized feature vector $\mathbf{y}$ will be scaled by a factor that is equal to the eSNR of the n^{th} variable.

$$\tilde{y}_n = y_n \cdot \text{eSNR}(y_n) \tag{4.14}$$

Those variables with large local variance (i.e. low approximate SNR) will be scaled down more. The weighted features $\tilde{\mathbf{y}}$ will be processed by PCA. Note that variable weighting is necessary before applying PCA because PCA does not differentiate

variable variance caused by noise (local) or system degradation (global). PCA may fail to capture the variance direction that is most relevant to system degradation. This is easy to understand because PCA disregards the timing information in a time series.

4.5.3 Principal Component Analysis

The weighted variables in $\tilde{\mathbf{y}}$ are potentially correlated with each other. Correlated variables may compromise the RUL prediction models and bias the prediction results. PCA is a common technique to transform potentially correlated variables into a smaller number of uncorrelated variables. PCA is defined [Jolliffe, 2002] as an orthogonal linear transformation that transforms the data to a new coordinate system such that the first coordinate (called the first principal component, or PC) represents the direction of the greatest variance of the original data, and the second coordinate/PC represents the direction of the second greatest variance that is orthogonal to the first PC, and so on.

The Karhunen-Loéve transform (KLT) [Jolliffe, 2002] is a common method to compute PCA (see Appendix A). In short, PCA applies the following transformation to all the samples $\tilde{\mathbf{y}}$ from both training and test instances:

$$\mathbf{z} = (z_1, z_2, \ldots, z_M)^T = V_M^T \cdot (\tilde{\mathbf{y}} - \bar{\tilde{\mathbf{y}}}) \tag{4.15}$$

where $\bar{\tilde{\mathbf{y}}}$ is the mean of $\tilde{\mathbf{y}}$, and V_M consists of the eigenvectors of the covariance matrix of $\hat{\mathbf{y}}$ corresponding to the first M PCs. The number of significant PCs to retain is decided by a threshold, e.g. 90%, on the percentage of total variance explained by the selected PCs. The resultant PCs form a new time series ${}^{l}Z_I$ for each training or testing instance, which are ready for TSBP modeling.

4.6 Beyond data preparation: issues of data collection

Up till now, the issue of data preparation for TSBP modeling has been discussed from the perspective of algorithm development, including the common operations such as denoising, feature extraction and selection, normalization, and so on. However, from the perspective of prognostics system design, the scope of data preparation can be further extended to the data collection stage, which directly affects the quality of the collected data. Data quality a crucial factor on the performance of a prognostics model and the success of a prognostics application.

Data quality, on the context of predictive analysis, is more than the signal quality, such as noise level, for each single measurement; it is also tightly related to the data collection plan. Unlike the classic subject of Design of Experiments (in biomedical applications, for instance), data collection in prognostics applications are not only necessary in the development phase (e.g. for test of hypothesis), but also indispensable in the deployed solution for an engineering system that operates in the field. The in-field condition will prevent the data collection plan from being fully controlled. Data collection designing for prognostics applications has to take into consideration the feasibility to carry out the plan with minimal interference to the normal operation of the system while trying to ensure the best data consistency.

RUL prediction requires trend analysis on the time series of the system's health condition. Consistency of data samples in the time series plays an important role for modeling. The TSBP model, as well as many other RUL prediction models, assumes that the data are collected in cycles consistently and the system's operation in each cycle causes comparable wear (see the assumptions of TSBP in Section 3.5). Data consistency can be addressed from the following two perspectives:

- Consistency of measurement timestamps

- Consistency of measurement condition

An engineering system may have a long life that makes continuous measurement impractical in most applications. Therefore the system may have to be instrumented at intervals, which leads to the concept of measurement timestamps. The measurement time stamp is a macro-level concept that has to be differentiated from the micro-level signal sampling frequency (e.g. for collecting vibration signal). For example, measurement of bearing vibration in a rotary machine may have a sample frequency of 10kHz, but the measurement may take place for only 2 seconds every hour. Here the measurement time stamp will be related to the 1-hour interval. In general, the timestamps of measurement can be one of the following:

- Calendar time, e.g. mm/dd/yyyy
- Operation time, e.g. hours in operation
- Operation cycle counter, e.g. 1,2,3,...

Calendar time can be used to timestamp the measurements if the system continuously operates without apparent interruption due to weekends or holidays, such as a wind turbine. With calendar time as the time stamp, it is assumed that the load of the system in average is comparable for each time period; variations in load will be treated as noise and will be handled during data processing. Sometimes the operation time of a system can be used as a more accurate time stamp for the measurements if the system operates with intermittent stops, such as a machine tool or equipment in the factory floor. Again, it is assumed that for each unit time, the system will perform a comparable level of work. For those systems operating in cycles, e.g. a product line producing discrete products, measurement can be carried out and timestamped by the number of operational cycles. Operational cycles provide the most consistency in the load to the system.

The measurement timestamps reflect a period of operation rather than a single moment. The exact measurement method during each time interval may have one of the following scenarios:

- Snapshot at an arbitrary moment within the period
- Snapshot at a moment of similar conditions within the period
- The averaged measurement over the whole period, possible from many operational conditions

The system may or may not operate under identical operational conditions during the entire measurement interval. If yes, a measurement snapshot may be taken at any moment within the period; this also applies for the collection of parameters related to the system setup, which may keep constant for the entire period. If the operational conditions changes over over time, it is desirable to have measurement snapshots at a moment with similar operational conditions. Such moment can be found relatively easily in a cycle-based production line but may be hard in the case of stochastically changing operational conditions, such as the case of a wind turbine. In the later case it may be necessary to record the operating conditions during the measurement as well. If the operating conditions are too complex to record precisely and completely, such as the case of a heavy mining machine operating in the field, the option is to record the statistics of the measurements and conditions (e.g. average, minimum, maximum, etc.) over the period.

In summary, data collection from a real system during its normal operation has to be designed properly to ensure that the data quality is as close to the modeling assumptions as possible. In a certain case, a specially designed test cycle can be forced inserted into the intervals of the system's normal operation, e.g. the so-called

Fixed Cycle Feature Test (FCFT) method used by Liao and Lee [2009], in order to ensure a consistent instrumentation condition.

5 Case Study: Turbofan Engine RUL Estimation

In this chapter, the developed TSBP approach will be validated using the turbofan engine degradation simulation data publicly available from NASA Ames Prognostics Data Repository [Saxena and Goebe, 2008].

5.1 Simulation setup and data description

Saxena et al. [2008b] conducted a series of run-to-failure simulations to study turbofan engine degradation. The simulation model was built on CMAPSS (Commercial Modular Aero-Propulsion System Simulation) developed at NASA Army Research Laboratory [Frederick et al., 2007]. CMAPSS is able to simulate the operation of an engine model of the 90,000 lb thrust class under variable operational conditions, including altitude, Mach number, sea-level temperature. In addition, the engine thrust can be controlled by the Throttle Resolver Angle (TRA) value, which can be treated as another operational condition as defined in Section 4.2. By modifying 13 health parameters in CMAPSS, such as the efficiency or flow modifiers of these components, the user can simulate the effects of faults and deterioration in any of the engine's five rotating components, including fan, LPC (Low-Pressure Compressor), HPC (High-Pressure Compressor), HPT (High-Pressure Turbine), and LPT (Low-Pressure Turbine), as shown in Fig. 5.1. CMAPSS also provides 58 variables for different outputs.

In the simulation conducted by Saxena et al. [2008b], 21 of the 58 output variables are recorded to simulate sensor measurements to the system, which include, for

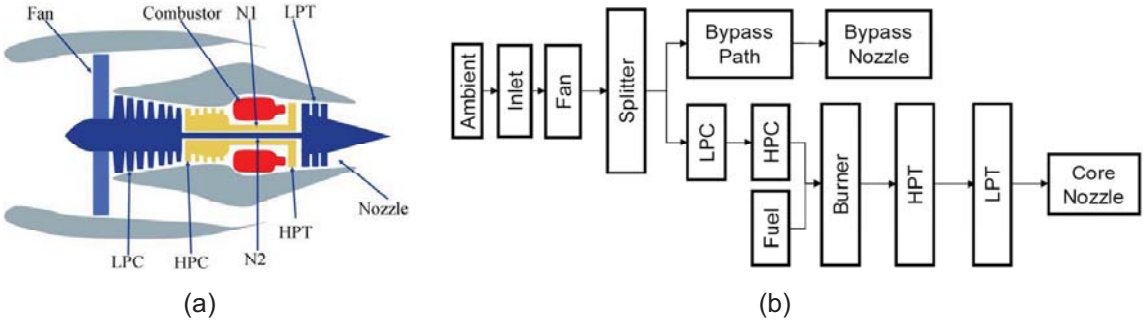

Figure 5.1: Structure of the turbofan engine simulated in CMAPSS [Frederick et al., 2007; Saxena et al., 2008b]. (a) Simplified engine diagram (b) Engine modules and their inter-connections

instance, temperature, pressure, and speed at various points in the system. Another 3 variables representing the engine operating conditions are recorded, namely altitude (kilo feet), Mach number (speed) and Throttle Resolver Angle (TRA) value (thrust setting). By running the simulation multiple times under different flight conditions (and thus engine conditions), data from multiple instances of the same engine model are collected. For each run of the simulation, the engine experiences complete run-to-failure operations, i.e., starting from brand new (with different degrees of initial wear and manufacturing variation), developing faults over a number of flights from one location to another, and finally reaching the failure condition measured by a set of predefined criteria. The engine life is defined as the number of total flight cycles in which the engine is in operation. Depending on various factors, the amount and rate of damage accumulation for each engine instance will be different, causing variable engine life. The recorded data for each engine instance provide a track of the engine's condition throughout its usage history in the format of a 24-dimensional time series (3 operating conditions and 21 sensor measurements for each flight cycle). During

each flight cycle only a snapshot of these 24 variables is taken to represent the engine condition for the current cycle. An excerpt of the multivariate time series from one instance is shown in table 5.1.

Table 5.1: Sample run-to-failure data from one engine instance

Cycle	Operating Setting 1	Operating Setting 2	Operating Setting 3	Sensor 1	Sensor 2	...	Sensor 21
1	42.0049	0.8400	100.0	445.00	549.68	...	6.3670
2	20.0020	0.7002	100.0	491.19	606.07	...	14.655
⋮	⋮	⋮	⋮	⋮	⋮		⋮
321	42.0058	0.8400	100.0	445.00	549.71	...	6.4590

Need to mention that the background information on the engine simulation model is not a must-to-know to perform RUL estimation using TSBP approach. Actually, when a data set created by this simulation was first provided to the 2008 PHM Data Challenge competition, the data was described as multivariate run-to-failure time series from multiple instances of an unspecified engineering system. The only information available was that the 24 dimensions for each cycle includes 3 condition variables and 21 sensor measurements. No failure mode information nor failure criteria was provided – for TSBP approach, they are not necessary though. TSBP is developed to circumvent the needs for explicit information on failure modes and failure criteria.

NASA has provided 4 data sets generated from 4 independent simulation experiments, each with different settings such as the number of engine operational

conditions, potential faulty components, and so on. Therefore only those instances in the same data set can be considered to from identical systems. Data sets 1 and 2 includes only one fault modes (HPC degradation) while data sets 3 and 4 includes two (HPC degradation and fan degradation). Data sets 1 and 3 include a single operational condition while data sets 2 and 4 includes six. Data set 4 represents the most complex case among the four. Summary of the 4 data sets are shown in Table 5.2.

Table 5.2: Experiment settings of the four data sets

	Data Sets			
	# 1	# 2	# 3	# 4
Number of fault modes	1	1	2	2
Number of operating conditions	1	6	1	6
Number of training units	100	260	100	249
Number of test units	100	259	100	248

Each data set is further divided into training and test subsets. The training set includes instances with complete run-to-failure data, which can be used to develop life prediction models. The test set includes instances with data up to a certain cycle prior to system failure, which are used for RUL estimation and algorithm performance evaluation. Note that the test instances are also simulated run-to-failure; even though only an earlier portion of the history is provided in the data set, the actual life of the test instances are still known – the ground true RUL of the test instances are also provided for prediction validation purposes. The actual failure mode for each training instance in data set 3 and 4 is not labeled.

In this study, the 4th data set is used to experiment and validate the proposed RUL estimation method. However, the provided test data set will not be used for algorithm validation, because it contains instances with incomplete run-to-failure data and thus not suitable for the performance evaluation method that relies on sequential RUL predictions at every time stamp throughout the instance's life. Therefore, in this study, the 249 instances in the training data set will be used for both training and testing. Out of the 249 instances, 99 are randomly selected and held as a validation set and the rest 150 are used for model development.

5.2 Performance metrics

An RUL prediction algorithm can be assessed from different perspective with various performance metrics. Selection of performance metrics has been a rapid-developing research topic. The traditional accuracy-based, precision-based and robustness-based performance metrics adopted from other prediction/forecasting or classification applications has limitations in providing a fair assessment to the algorithm's true merits and drawback in the context of prognostics applications. Therefore, much research in the society of PHM has been put on the developing of a systematic approach for performance evaluation of RUL prediction algorithm.

Inspired by the performance evaluation framework proposed by Saxena et al. [2010], four performance metrics are defined here, either directly adopted or with modifications. These metrics are defined on a sequence of RUL predictions for each instance, i.e. multiple predictions made at different time along the history of the instance with provided historical data up to that time. The earliest time at which a prediction is made is noted as t_p, meaning the start of RUL prediction routine, and the last time is noted as t_{EoUP}, meaning the End of Useful Predictions.

5.2.1 Prediction Horizon

Prediction Horizon (PH) is defined as the first RUL prediction that satisfies the α-bound criteria:

$$\text{PH} = t_E - t_{i_\alpha} \tag{5.1}$$

where i_α is the index of the first RUL prediction that satisfies the α-bound criteria. For a RUL prediction model that gives point estimation of RUL r_i at each time stamp t_i, the α-bound criteria evaluates whether r_i falls within the α-bound of the true RUL $r_i^*(\equiv t_E - t_i)$, as illustrated in Fig. 5.2:

$$t_{i_\alpha} = \min\{t_i | t_i \in [t_P, t_{EoUP}], r_i^* - \alpha \cdot t_E \leq r_i \leq r_i^* + \alpha \cdot t_E\} \tag{5.2}$$

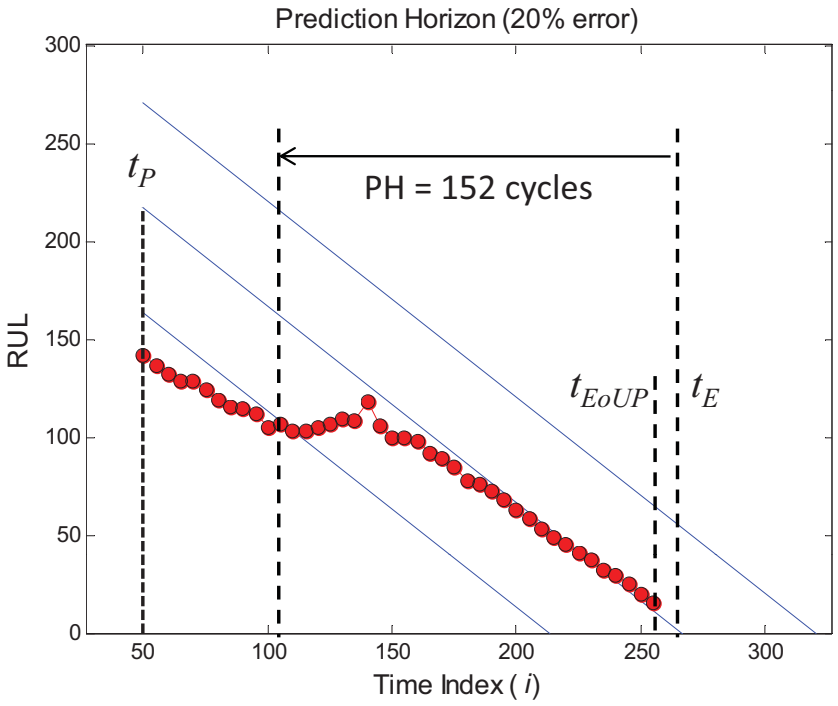

Figure 5.2: Prediction Horizon

For a RUL prediction model that provides density estimation of RUL $\pi(r_i)$, the α-bound criteria evaluates whether the cumulative probability of the predicted RUL within the α-bound of the true RUL is greater than a parameter $\beta(0 < \beta < 1)$:

$$t_{i_\alpha} = \min\{t_i | t_i \in [t_P, t_{EoUP}], \int_{r_i^* - \alpha \cdot t_E}^{r_i^* + \alpha \cdot t_E} \pi(r_i) dr_i \geq \beta\} \tag{5.3}$$

Note that PH defined for density estimation (with a β larger than 0.5, for instance) are more robust and more conservative than that for point estimation. PH given by Eq.(5.2) is usually smaller than PH given by Eq.(5.3). Therefore the PH's computed with these two methods are not comparable. One has to use the point-estimation method to compare two algorithms if one of them gives density-estimation and the other does not.

By definition, the PH metric does not prevent predictions made within PH to jump out of the α-bound; therefore additional performance metrics should be used to further evaluate whether the algorithm satisfies other requirements [Saxena et al., 2010].

5.2.2 Rate of Acceptable Predictions

Rate of Acceptable Predictions (AP) evaluates the rate of the predictions for all $t_i \geq t_H$ that fall into a cone-shape region of acceptable prediction errors, as illustrated in Fig. 5.3. The threshold t_H can be chosen as PH computed above or a certain specified value. This metric is defined here as an alternative to the $\alpha - \lambda$ performance metric defined by Saxena et al. [2010].

$$\text{AP} = \text{Mean}(\{\delta_i | t_H \leq t_i \leq t_{EoUP}\}) \tag{5.4}$$

For point estimation of RUL, δ_i is defined as follows

$$\delta_i = \begin{cases} 1 & \text{if } (1-\alpha)r_i^* \leq r_i \leq (1+\alpha)r_i^* \\ 0 & \text{otherwise} \end{cases} \tag{5.5}$$

For density estimation of RUL, δ_i is defined as follows

$$\delta_i = \begin{cases} 1 & \text{if } \int_{(1-\alpha)r_i^*}^{(1+\alpha)r_i^*} \pi(r_i)dr_i \geq \beta \\ 0 & \text{otherwise} \end{cases} \tag{5.6}$$

While PH is defined based on a uniform α-bound given by $\pm\alpha t_E$, AP is defined with a shrinking cone-shape α-bound given by $\pm\alpha t_i^*$. Therefore AP applies a more strict requirement to the prediction errors.

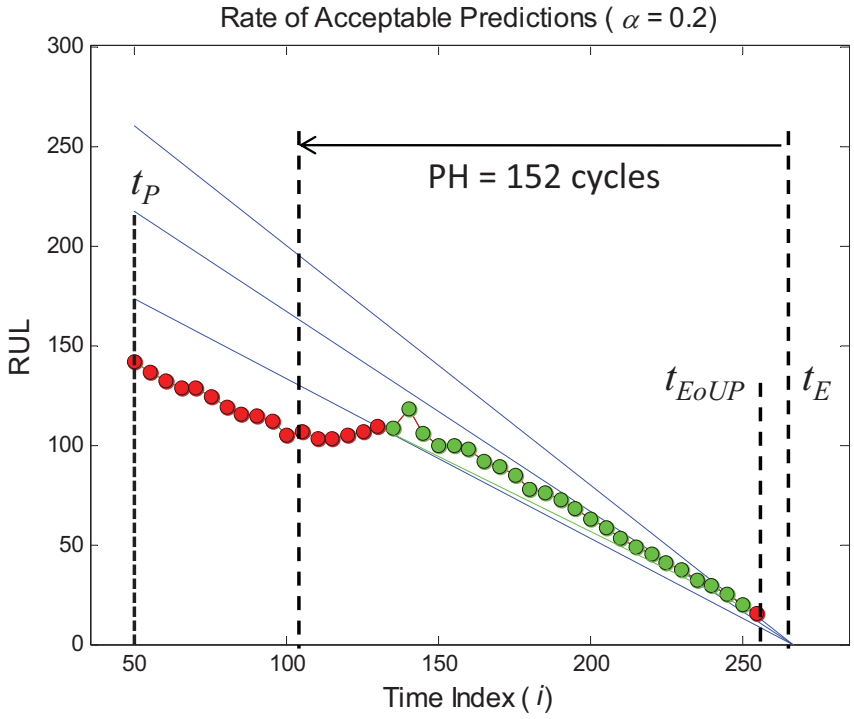

Figure 5.3: Rate of Acceptable Predictions

5.2.3 Relative accuracy

Relative accuracy (RA) evaluates the mean absolute percentage errors for all $t_i \geq t_H$. Compared with the AR metric, RA provides a quantitative measure of the prediction accuracy within the specified RUL.

$$\text{AR} = 1 - \text{Mean}(\{\frac{|r_i - r_i^*|}{r_i^*} | t_H \leq t_i \leq t_{EoUP}\}) \tag{5.7}$$

This metric is defined for point estimation of RUL only. For density estimation, a central tendency point estimate should be computed first Saxena et al. [2010] to get r_i. In this study, the point with 50% cumulative probability is used as the point estimate of RUL.

5.2.4 Convergence

Convergence (CG) evaluate how fast the performance of the predictions improves over time when more historical data become available. Give a performance metric M that can be evaluated at each time stamp t_i, CG is defined as

$$\text{CG} = 1 - \left(\frac{\frac{1}{2}\sum_{i=P}^{EoUP}(t_{i+1}^2 - t_i^2)M_i}{\sum_{i=P}^{EoUP}(t_{i+1} - t_i)M_i} - t_P \right) \cdot \frac{1}{t_{EoUP} - t_P} \tag{5.8}$$

The CG metric has a value between 0 and 1. If M_i decreases as performance improves (e.g. mean absolute error, standard deviation of errors, etc.), then a value larger than 0.5 for CG will indicate converging for the prediction; the larger CG is, the faster the convergence is. However, if M_i increases as performance improves (e.g. the AP and RA performance), then a value smaller than 0.5 for CG will indicate converging; in such case it is better to define CG with a slight change to ensure consistency of CG value:

$$\text{CG} = \left(\frac{\frac{1}{2}\sum_{i=P}^{EoUP}(t_{i+1}^2 - t_i^2)M_i}{\sum_{i=P}^{EoUP}(t_{i+1} - t_i)M_i} - t_P \right) \cdot \frac{1}{t_{EoUP} - t_P} \tag{5.9}$$

In this case study, Absolute Error is chosen as the performance metric M for calculating CG.

5.2.5 Performance score for parameter optimization

To evaluate the algorithm performance based on multiple prediction series from K test instances, the median of the K performance scores for each metric will be used:

$$\begin{aligned} \text{PH} &= \text{Median}(\{^k\text{PH}\}_K) \\ \text{AP} &= \text{Median}(\{^k\text{AR}\}_K) \\ \text{RA} &= \text{Median}(\{^k\text{RA}\}_K) \\ \text{CG} &= \text{Median}(\{^k\text{CG}\}_K) \end{aligned} \tag{5.10}$$

An objective function is necessary to compare the performance of different models, especially during model parameter optimization. Among the four metrics, PH has a unit of time while the others have a value between 0 and 1 (with 1 meaning perfect). PH will be used as preliminary requirement for the performance while a weighted sum of the other three will be used as the objective:

$$\text{maximize} \qquad w_1 \cdot AP + w_2 \cdot RA + w_3 \cdot CG \tag{5.11}$$

where the weights w_1, w_2, w_3 should be chosen, ideally, based on the design requirement of the prognostics algorithm, which is not defined for the current case study. Here the weights are set as follows, with more emphasis on AP, a little less on RA and the least on CG:

$$w_1 = 0.6, w_2 = 0.3, w_3 = 0.1 \tag{5.12}$$

5.3 TSBP modeling, prediction and evaluation

5.3.1 Model training

As the raw data provided are already in the form of multivariate time series, no feature extraction procedures are necessary. However the raw data are collected from 6 operational conditions/regimes, i.e. the data for different cycles may be collected under any one of the 6 regimes. As shown in Fig. 5.4. The sensor readings can hardly show any trend for the life cycle of the system if the operating regimes are not differentiated; within one operating regime, the sensor readings do show a rising trend. In this study, the multi-regime data normalization method described in Section 4.5 is employed to preprocess the raw data so that the effect of operating regimes can be removed.

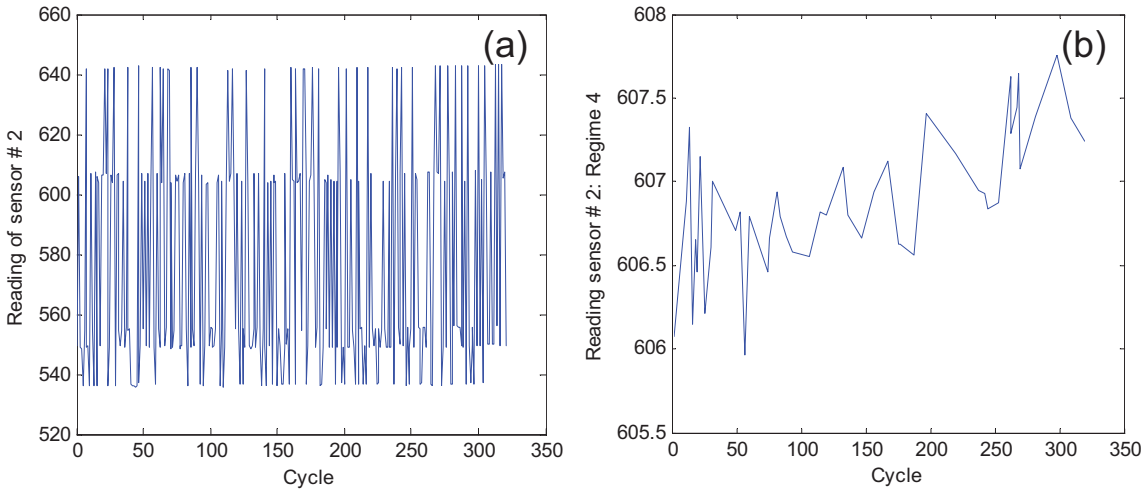

Figure 5.4: Raw data of sensor No. 2 from one training instance (a) All operating regimes together (b) Operating regime 4 only

The three variables for operational conditions are used to cluster the operating

regimes. Table 5.3 shows the cluster center and radius of the operating regimes obtained from a simple k-means clustering algorithm. It shows that the six operating regimes are far separated and can be considered as discrete - this will greatly simplify the processing of data normalization.

Table 5.3: Six operating regimes: cluster center and radius

Cluster center			Cluster radius
Condition 1	Condition 2	Condition 3	
42.0030	0.8405	100.0000	0.0030
35.0030	0.8405	100.0000	0.0030
0.0015	0.0005	100.0000	0.0011
20.0030	0.7005	100.0000	0.0030
25.0031	0.6205	60.0000	0.0030
10.0030	0.2505	100.0000	0.0030

Then each sensor will be normalized individually using the sample mean and variance in each operating regime (see Section 4.5.1). A few of the normalized sensors from selected training instances are shown in Fig. 5.5, where each row is for one instance and each column is for one sensor. It shows that the first two sensors (No. 2 and 8) exhibit consistent rising trend in different instances while the rest exhibit two different trends, which can be possibly interpreted as the effect of the two fault modes in the data set.

With the multi-regime data normalization method, sensor weighting is employed instead of sensor selection. However, a rough screening of the given 21 sensors are still necessary to exclude binary- or constant-value sensors for TSBP modeling. All the

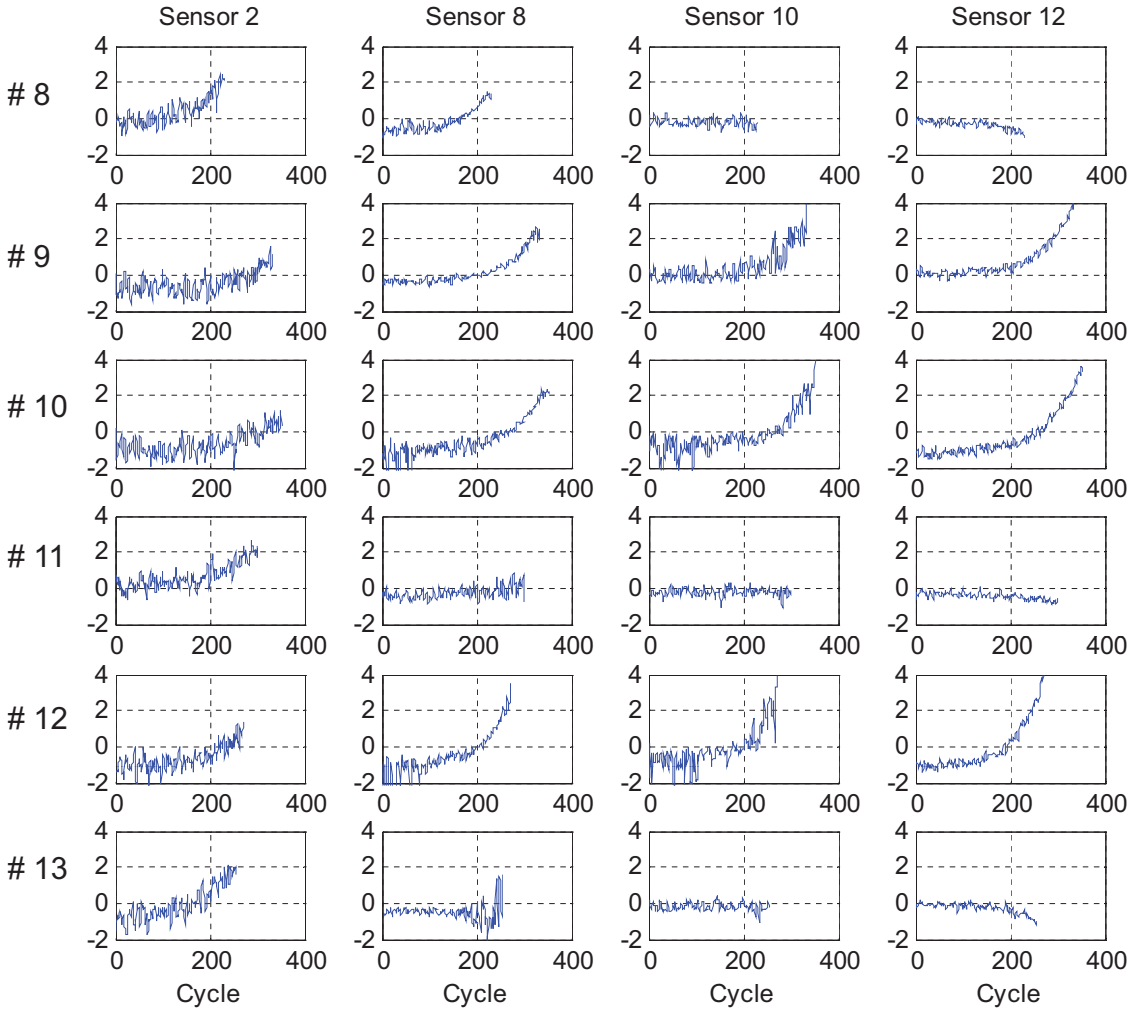

Figure 5.5: Normalized sensor data of selected training instances

identified 13 continuous-value sensors, namely No. 2, 3, 4, 7, 8, 9, 11, 12, 13, 14, 15, 20 and 21 are included for further processing. Then, the variable weighting method described in Section 4.5.2 is applied to differentiate the contribution of relevant sensors from irrelevant sensors. Table 5.4 shows the weights for the selected 13 sensors using the sensor's eSNR.

Then PCA is applied to remove the linear correlations among the variables.

Table 5.4: Empirical Signal/Noise Ratio

Sensor index	2	3	4	7	8	9	11
eSNR	0.7314	0.7337	0.8364	0.8959	0.7533	0.8936	0.8744
Sensor index	12	13	14	15	20	21	
eSNR	0.9048	0.7549	0.9009	0.9132	0.7003	0.6986	

The number of significant PCs to retain is decided by a threshold on the minimal percentage of total variance explained by the selected PCs. The threshold is set to 90% and three PCs ($M = 3$) are retained. The samples from training and test instances are then transformed to the PC domain using Eq. (4.15). During degradation trajectory abstration, the abtained PC series will be smoothed using kernel regression; the resultant time series are recorded as non-parametric degradation models ${}^{l}G$, each from one training instances. A few examples of the PC series and the corresponding degradation models (smoothed trajectories) are shown in Fig. 5.6.

5.3.2 RUL estimation

During testing, for each test instance, multiple RUL predictions will be made at different cycles along the instance's life; each prediction is made based on the up-to-date data till the corresponding time Prediction are made between cycle t_P and t_{EoUP} at each time stamp spaced with interval of Δt. The following choices are used in this case study:

$$t_p = 50, t_{EoUP} = t_E - 10, \Delta t = 5$$

The local RUL predictions given by individual degradation models is forced to be constrained to the limit of 350 cycles. Any local predictions longer than 350 cycles

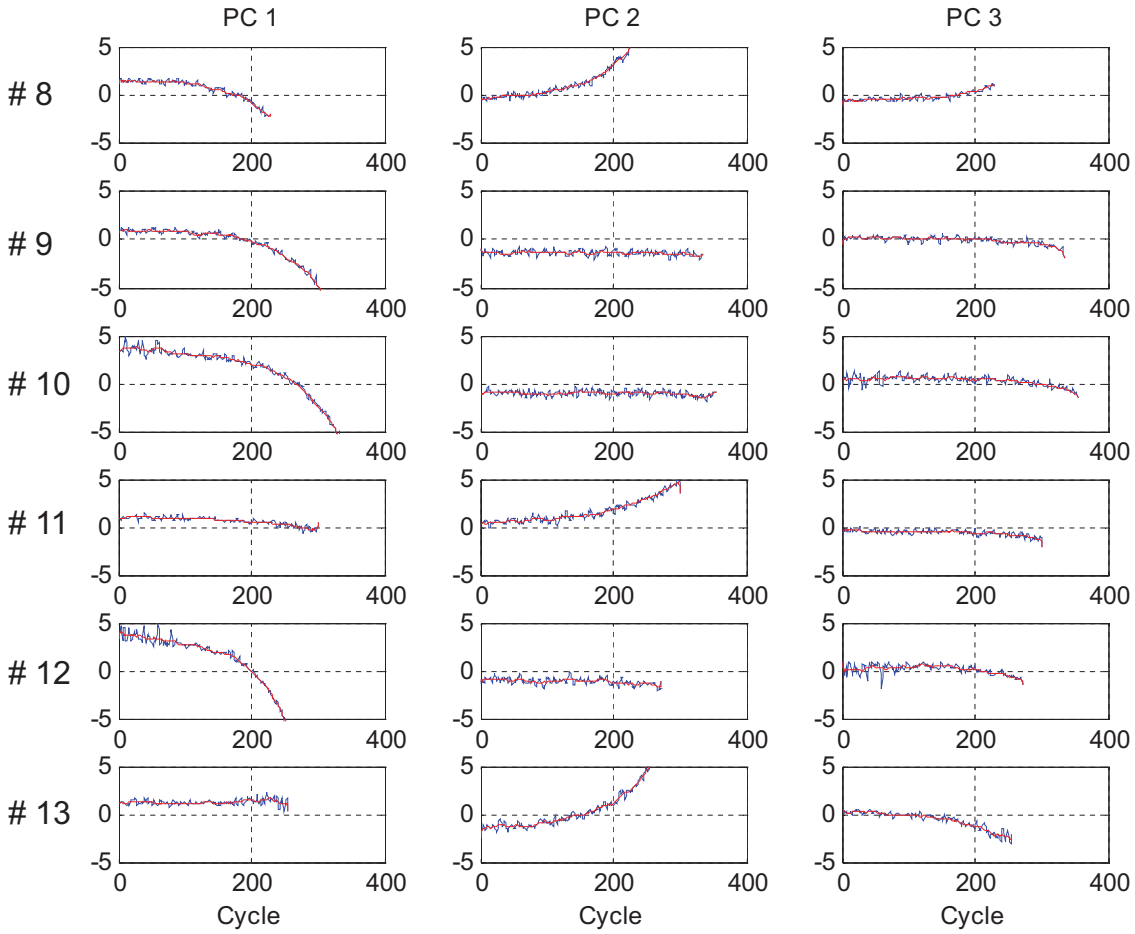

Figure 5.6: PC trajectories and the smoothed PC trajectories from selected training instances

will be treated as prediction outliers and will not be included in RUL aggregation. This adjustment help to reduce the predicted uncertainty bound produced by TSBP.

Two examples of RUL predictions are shown in Fig. 5.7. Both cases show the 90% confidence interval and the central tendency of the RUL estimate. Case (a) shows desirable performance, where the predictions converge to the true RUL as time increases, whereas case (b) demonstrates undesirable performance.

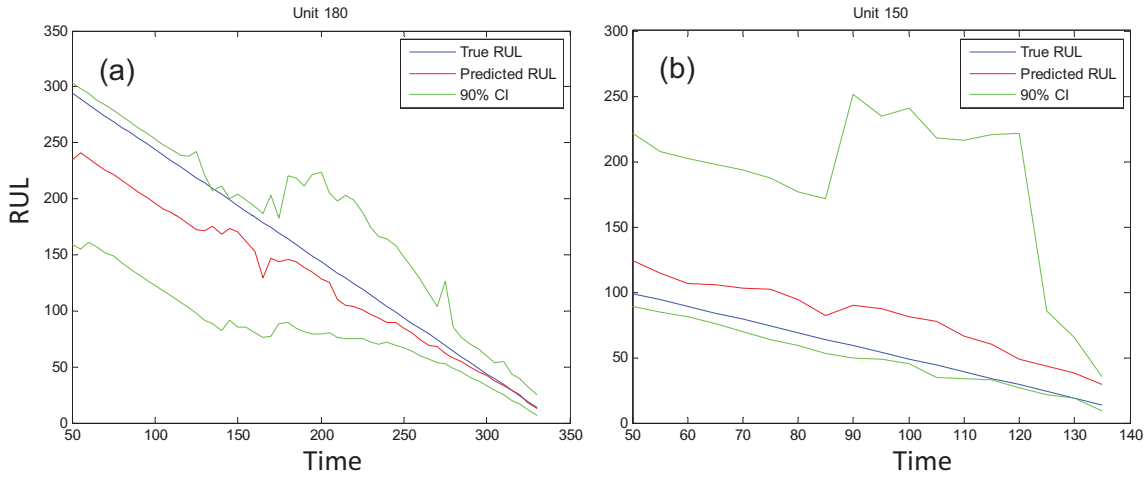

Figure 5.7: Trajectory of RUL predictions vs. Time (a) Predicted RULs converge to the actual RUL (b) Predicted RULs show a large bias towards the end of life

5.3.3 Model tunning

5.3.3.1 TSBP model settings

Under the framework of TSBP approach for RUL estimation, there are more than one option for the distance definition and more than one free-choice parameters in the four procedures. These options or parameters are summarized in Table 5.5; the method to decide the parameters are also included.

5.3.3.2 Method for model parameter tuning

Two parameters, the kernel width ρ for degradation trajectory abstraction and the spread ratio γ for distance evaluation, have to be decided for an TSBP model. In this study, the n-fold cross-validation method is used to compare the model performance with different settings. The procedures for cross-validation are described as follows:

Table 5.5: Options in TSBP modeling

Procedure	Algorithm	Parameters	How to decide parameters
Data preparation	Multi-regime data normalization	Variables to include	Include all; apply variable weighting (Eq. (4.14))
Degradation trajectory abstraction	Kernel regression	Kernel width ρ (Eq.(3.9))	Cross-validation (See below)
Similarity evaluation	MED-TL	Spread ratio γ (Eq.(3.15))	Cross-validation (See below)
	MED-DA	Spread ratio γ (Eq.(3.15))	Cross-validation (See below)
	MED-TL-DA	Spread ratio γ (Eq.(3.15))	Cross-validation (See below)
Model aggregation	Kernel density estimation	Bandwidth h (Eq.(3.25))	Adaptive algorithm (Section 3.4.3)

1. The samples(training instances) are randomly divided into n mutually exclusive subsets of approximately equal size, called n folds.
2. Specify a model setting;
3. Each time one out of the n folds are held for test and the remaining $n-1$ folds are used for training;
4. Repeating step (3) n times until each of the n folds has been used for test once;
5. Performance metrics are applied to the prediction results at each iteration, and

the overall performance across n iterations is used as the performance score for the model with the chosen setting;

6. Repeat step (2–5) for all the model setting to be tested.

In this study, 5-fold cross-validation is applied. For each cross-validation experiments, the four performance metrics described in Section 5.2, namely PH, AP, RA and CG, will be computed. The total performance score, $0.6 \times \text{AP} + 0.3 \times \text{RA} + 0.1 \times \text{CG}$, will be used as the final measure to rank model performance.

As experimented in this case study, one round of cross-validation (150 instances, each including multiple RUL predictions) may take up to hours to finish while using the TSBP approach. Due to this high computational load, the model parameters will not be easily optimized using classical search methods. Therefore, only a number of pre-selected values for the parameters will be tested. In addition, the two parameters will not be optimized simultaneously, but rather in sequence, in order to further reduce the computational load. The sequence to decide the two parameters will follow the reverse order of the data processing procedures. In other words, the spread ratio γ for distance evaluation will be tuned first; and then the kernel width ρ for degradation trajectory abstraction will be tuned later with the spread ratio γ fixed with the selected best value.

In this study, the model performance with three different settings of sensor subsets will be compared too. The purpose of the comparison is to demonstrate how the selection of sensor subset affects the model performance, but not intended to optimize the selection.

5.3.3.3 Tuning spread parameter for distance evaluation

Cross-validation experiments are conducted with a few choices of spread parameter γ while fixing the choice of other options and parameters. The settings of TSBP model

are summarized in Table 5.6.

Table 5.6: Tuning spread parameter for distance evaluation

Procedure	Algorithm and Parameters
Data preparation	Multi-regime data normalization with variables No. 2, 3, 4, 7, 8, 9, 11, 12, 13, 14, 15, 20 and 21
Degradation trajectory abstraction	Kernel regression with kernel width $\rho = 10$
Similarity evaluation	MED-TL with γ=0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.75, 1,∞
	MED-DA with γ=0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.75, 1, ∞
	MED-TL-DA with γ=0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.75, 1, ∞
Model aggregation	Kernel density estimation with adaptive selection of bandwidth h

The performance evaluation results for TSBP with the three distance definitions and different spread parameters are shown in Table 5.7, Table 5.8 and Table 5.9 respectively. Based on the total performance score, we can see that the best setting is the MED-DA distance with spread ratio $\gamma = 0.3$.

Table 5.7: Algorithm performance with MED-TL distance and different spread parameters

Performance Metrics	Spread ratio γ for distance evaluation								
	0.1	0.2	0.3	0.4	0.5	0.6	0.75	1	∞
PH: Prediction Horizon ($\alpha = 0.2, \beta = 0.8$)	82.0	81.5	84.0	81.0	79.0	80.5	77.5	75.5	78.5
AP: Rate of Acpt. Pred. ($t_H = 80, \alpha = 0.2$)	0.5714	0.5167	0.3571	0.3571	0.3452	0.3095	0.2857	0.2857	0.2857
RA: Relative Accuracy ($t_H = 80$)	0.7850	0.7633	0.7414	0.7221	0.7129	0.7191	0.7114	0.7054	0.7044
CG: Convergence ($t_H = 80$)	0.6105	0.6189	0.6130	0.6029	0.6037	0.5983	0.6025	0.5997	0.6041
Total: 0.6 AP+0.3 RA+0.1 CG	**0.6394**	0.6009	0.4980	0.4912	0.4814	0.4613	0.4451	0.4430	0.4432

Table 5.8: Algorithm performance with MED-DA distance and different spread parameters

Performance Metrics	Spread ratio γ for distance evaluation								
	0.1	0.2	**0.3**	0.4	0.5	0.6	0.75	1	∞
PH: Prediction Horizon ($\alpha = 0.2, \beta = 0.8$)	90.5	91	94	93	93	93	96	93	92
AP: Rate of Acpt. Pred. ($t_H = 80, \alpha = 0.2$)	0.5000	0.5524	0.5714	0.5714	0.5714	0.5714	0.5714	0.5524	0.5714
RA: Relative Accuracy ($t_H = 80$)	0.7853	0.7958	0.7929	0.7889	0.7849	0.7904	0.7798	0.7771	0.7752
CG: Convergence ($t_H = 80$)	0.6183	0.6301	0.6138	0.6169	0.6032	0.6010	0.5986	0.6025	0.5988
Total: 0.6 AP+0.3 RA+0.1 CG	0.5974	0.6332	**0.6421**	0.6412	0.6386	0.6401	0.6038	0.6248	0.6353

Table 5.9: Algorithm performance with MED-TL-DA distance and different spread parameters

Performance Metrics	Spread ratio γ for distance evaluation								
	0.1	0.2	0.3	**0.4**	0.5	0.6	0.75	1	∞
PH: Prediction Horizon ($\alpha = 0.2, \beta = 0.8$)	113.5	122	115	119.5	119	123.5	121	120	119.5
AP: Rate of Acpt. Pred. ($t_H = 80, \alpha = 0.2$)	0.2143	0.2143	0.2143	0.2405	0.2143	0.2143	0.2143	0.2143	0.2071
RA: Relative Accuracy ($t_H = 80$)	0.6253	0.6511	0.6575	0.6657	0.6351	0.6291	0.6198	0.6421	0.6182
Convergence ($t_H = 80$)	0.6021	0.5813	0.5794	0.5696	0.5631	0.5577	0.5475	0.5534	0.5461
Total: 0.6 AP+0.3 RA+0.1 CG	0.3764	0.3820	0.3838	**0.4010**	0.3754	0.3731	0.3693	0.3766	0.3644

Table 5.10: Algorithm performance with different kernel width for degradation trajectory abstraction

Performance Metrics	Kernel width ρ for degradation trajectory abstraction								
	3	5	**7**	10	15	20	25	35	50
PH: Prediction Horizon ($\alpha = 0.2, \beta = 0.8$)	118.5	104	100.5	94	92	82.5	78.5	62.5	46
AP: Rate of Acpt. Pred. ($t_H = 80, \alpha = 0.2$)	0.6214	0.6214	0.6214	0.5714	0.4476	0.3786	0.2857	0.2143	0.1429
RA: Relative Accuracy ($t_H = 80$)	0.7983	0.8005	0.7980	0.7929	0.7637	0.7277	0.7029	0.6400	0.5616
Convergence ($t_H = 80$)	0.6028	0.6128	0.6278	0.6138	0.5975	0.6099	0.6095	0.6120	0.5955
Total: 0.6 AP+0.3 RA+0.1 CG	0.6726	0.6743	**0.6750**	0.6421	0.5574	0.5064	0.4432	0.3818	0.3138

5.3.3.4 Tuning kernel width for degradation trajectory abstraction

With the distance evaluation settings fixed to the best choice (MED-DA distance with $\pi = 0.3$) from the previous tuning step, the kernel width ρ for degradation trajectory abstraction will be further tuned through cross-validation experiments. Experiments with the following choices of ρ are conducted: $\rho = 3, 5, 7, 10, 15, 20, 25, 35, 50$. The algorithm performance for these choices of ρ are summarized in Table 5.10. It shows that with $\rho = 7$ and the previously optimized spread ratio $\gamma = 0.3$, the algorithm yields the best performance.

5.3.3.5 Performance comparison with different sensor subsets

As mentioned before, sensor selection is not optimized in this case study, but instead, variable weighting is applied. Another three experiments are conducted to compare the algorithm performance using different sensor subset rather than sensor weighting.

- Exp 1: Use the same sensor set as before but without applying variable weighting .
- Exp 2: Uses sensor No. 2, 3, 4, and 11, which exhibit consistent monotonic trend (either rising or falling as cycle increases) for both failure modes. These sensors can best linearly classify early-life from end-life conditions.
- Exp 3: Uses sensor No. 12, 14 and 15, which are the top ranking variables in terms of eSNR as discussed in Section 4.5.2. The value of eSNR can be found in Table 5.4.

All three experiments use the MED-DA distance definition and the optimal parameters $\gamma = 0.3, \rho = 7$ from previous tuning procedures. Table 5.11 summarizes the cross-validation performance scores for these three experiments as well as those

Table 5.11: Algorithm performance with different sensor subsets without variable weighting

Performance Metrics	Experiments			
	Original	Exp1	Exp2	Exp3
PH: Prediction Horizon ($\alpha = 0.2, \beta = 0.8$)	100.5	93	68.5	82
AP: Rate of Acpt. Pred. ($t_H = 80, \alpha = 0.2$)	0.6214	0.5167	0.4667	0.4667
RA: Relative Accuracy ($t_H = 80$)	0.7980	0.7880	0.7932	0.7562
Convergence ($t_H = 80$)	0.6278	0.6281	0.6383	0.6141
Total: 0.6 AP+0.3 RA+0.1 CG	**0.6750**	0.6092	0.5818	0.5683

for the original algorithms with sensor weighting. It shows that sensor weighting method proposed in this thesis outperforms all the other three scenarios of sensor selection.

5.3.4 Performance evaluation using the validation set

The best model settings obtained from the aforementioned cross-validation experiments are summarized in Table 5.12.

With these settings, TSBP model are retrained using all 150 training instances and then validated using 99 test instances in the validation set (refer to the data division method described in Section 5.1. The performance metrics evaluated for the algorithm during cross-validation and final validation are listed in Table 5.13. In addition to the four performance metrics discussed above, a few traditional metrics are also evaluated for reference purposes. These metrics include Mean of Bias, Mean Absolute Errors, Root Mean Squared Errors, Symmetric Mean Absolute Percentage Errors, Standard Deviation of Errors and Median of absolute Errors (see Section 2.5.2

Table 5.12: Best TSBP settings

Procedure	Algorithm and Parameters
Data preparation	Multi-regime data normalization with variables No. 2, 3, 4, 7, 8, 9, 11, 12, 13, 14, 15, 20 and 21
Degradation trajectory abstraction	Kernel regression with kernel width $\rho = 7$
Similarity evaluation	Distance definition MED-DA with spread ratio$\gamma = 0.3$
Model aggregation	Kernel density estimation with adaptive selection of bandwidth h

for definitions). It shows that the performance of the algorithm improved during final validation over cross-validation. This can be attributed to the increased number of training instances that expand the library of degradation patterns.

5.4 Benchmarking with the Neural Network approach

There are many different approaches to constructing Neural Network (NN) models for RUL estimation, especially the setting of inputs and outputs for an NN. And then, the type of NN has to be specified, such as Feed-Forward Back-Propagation NN, Recurrent NN, Radial Basis Function Network (RBFN), and so on; then the detailed internal structure has to be decided, including the number of layers and neurons, transfer functions at each layer, etc.; in some cases, a certain free parameter has to be decided too, such as the spread parameter for an RBFN.

Table 5.13: RUL estimation performance using TSBP approach

Performance Metrics	Cross-validation	Final validation
PH: Prediction Horizon ($\alpha = 0.2, \beta = 0.8$) (density estimation)	100.5	104
PH*: Prediction Horizon ($\alpha = 0.2, \beta = 0.8$) (point estimation)	168.5	158
AP: Rate of Acpt. Pred. ($t_H = 104, \alpha = 0.2$)	0.6214	0.7368
RA: Relative Accuracy ($t_H = 80$)	0.7980	0.8320
CG: Convergence ($t_H = 80$)	0.6278	0.6167
Total: $0.6 \times AP + 0.3 \times RA + 0.1 \times CG$	0.6750	0.7534
MBias: Mean of Bias/Errors	2.4799	-2.0762
MAE: Mean Absolute Errors	21.0848	20.7676
RMSE: Root Mean Squared Errors	30.1945	31.9886
sMAPE: Symmetric Mean Abs. Pct. Errors	0.2302	0.2292
S: Standard Deviation of Errors	30.0956	31.9262
MAD: Median of absolute Errors	20.9622	20.7587

5.4.1 Design of NN for RUL prediction

In this case study, N inputs and one output for an NN are defined as follows:

- *Inputs*: N equally-spaced health indices (with a space of k cycles) in the history up to the current time stamp, $(z_{i-(N-1)k}, z_{i-(N-2)k}, ..., z_i)$
- *Outputs*: the current RUL, $r_i = t_E - t_i$

The original multivariate data $\mathbf{x}_i$ has to be converted into 1-D data z_i first before it can be used as the NN inputs. In this case study, the multi-regime health assessment method described in Section 4.4 is used to compute the health index series. The sensors to include for analysis are those with consistent trend, i.e. sensor No. 2, 3, 4, and 11. Linear models in the form given by Eq.(4.7) are used as local health assessment models. A few examples of the obtained 1-D health index series after health assessment are shown in Fig. 5.8 (blue curves) The health index series will

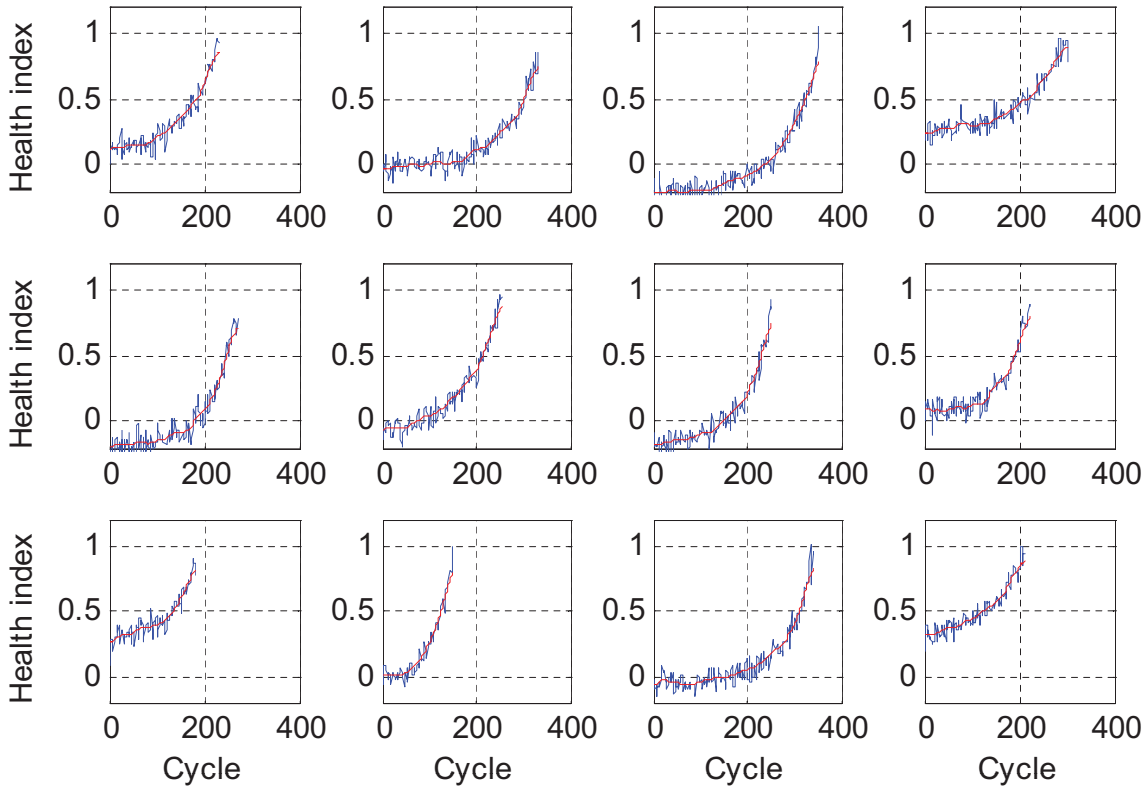

Figure 5.8: Health index series and the degradation trajectory (smoothed health indices) of 12 training instances

be smoothed (Fig. 5.8, red curves) using kernel regression (with kernel width $\rho = 7$)

before being used to construct the inputs for the RBFN. The number of NN inputs is set to $N = 17$ and the spacing is set to $k = 3$.

The type of NN used here is RBFN. RBFN has one hidden layer and one output layer. The hidden layer has a number of Radial Basis Function (RBF) neurons, which calculate the distances between the input vector and the neurons based on RBF:

$$f_{RBF}(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - c\|^2}{2\rho_{RBF}^2}\right) \tag{5.13}$$

where ρ_{RBF} is the RBF spread parameter. The output layer has a linear neuron, which calculate a weighted sum of the hidden layer outputs. Due to the inclusion of RBF neurons, RBFN is capable of capturing the similarity between a test input and the training inputs to make predictions; the usage of "similarity" in RBFN, of course, is much simpler than that in the TSBP approach.

In order to train the RBNN model, the training samples has to be prepared/constructed using the health index series from L training instances, with multiple training samples generated from each instance by sliding a time window of size $k \cdot N$ over the time series:

Inputs	Outputs	
$(z_1, z_{1+k}, ..., z_{1+(N-1)k})$	$t_E - t_{1+(N-1)k}$	} constructed from the l^{th} instance
$(z_2, z_{2+k}, ..., z_{2+(N-1)k})$	$t_E - t_{2+(N-1)k}$	
$\vdots$	$\vdots$	
$(z_{E-(N-1)k}, z_{E-(N-2)k}, ..., z_E)$	0	

During training, the RBFN will start from a single neuron. The model Mean Squared Error (MSE) will be computed. If MSE is greater than a specified goal, a new RBF neuron will be added with weights equal to the input vector that produces the greatest

error, and then the network will be retrained. This process will repeat until the goal of MSE is reached or the maximum number of neurons is reached. In this case study, the RBFN has difficulty to converge to a small MSE (e.g. a square of 20 cycles) due to the big variation in the data set. Therefore, the training samples are narrowed down to those with target RULs less than 200 cycles. And the maximum number of neurons is set to 50 to force early stop for over-fitting prevention.

RBFN has a free-choice parameter, the RBF spread, which is optimized through cross-validation. The final choice of RBF spread is 0.65.

5.4.2 Performance evaluation and analysis

The final settings of RBFN are summarized in Table 5.14. The RUL prediction performance is shown in Table 5.15, including the performance from cross-validation (using only the 150 training instances), the performance from the final validation (using 150 training instances and 99 test instances), and the performance of TSBP in final validation. Sample predictions for a few test instances in the validation set are shown in Fig. 5.9. The prediction errors vs. the true RUL, regardless timestamps and instances, are shown in Fig. 5.10. The TSBP approach has demonstrated higher accuracy and prediction consistency than RBFN.

Table 5.14: Settings of RBF Network for RUL estimation

Smoothing kernel width ρ	7
Number of NN inputs N	13
Input construction spacing k	3
Number of RBF neurons	50
RBF spread parameter ρ_{RBF}	0.65

Table 5.15: RUL estimation performance using RBF Network

Performance Metrics	Cross validation	Final validation	TSBP Final
PH*:Prediction Horizon ($\alpha = 0.2, \beta = 0.8$) (point estimation)	178.5	177	158
AP: Rate of Acpt. Pred. ($t_H = 80, \alpha = 0.2$)	0.4286	0.4000	0.7368
RA: Relative Accuracy ($t_H = 80$)	0.7240	0.7124	0.8320
CG: Convergence ($t_H = 80$)	0.6406	0.6137	0.6167
Total: $0.6 \times AP + 0.3 \times RA + 0.1 \times CG$	0.5384	0.5151	0.7534
MBias: Mean of Bias/Errors	10.5995	6.2551	-2.0762
MAE: Mean Absolute Errors	23.7003	23.5003	20.7676
RMSE: Root Mean Squared Errors	31.9915	34.6383	31.9886
sMAPE: Symmetric Mean Abs. Pct. Errors	0.2631	0.2523	0.2292
S: Standard Deviation of Errors	30.1876	34.0742	31.9262
MAD: Median of absolute Errors	22.5228	23.0269	20.7587

A few comments on the RBFN approach are listed below:

- This RBFN approach produces only point estimation of RUL, therefore the point-estimation based PH metric is used, which gives a much longer PH value than a density-estimation based version.
- Predictions with exceptional large errors (e.g. over 1000) can be found occasionally, which can be attributed to the over-fitting problem of the NN. The performance results shown in Table 5.15 has been computed after removing these prediction outliers.

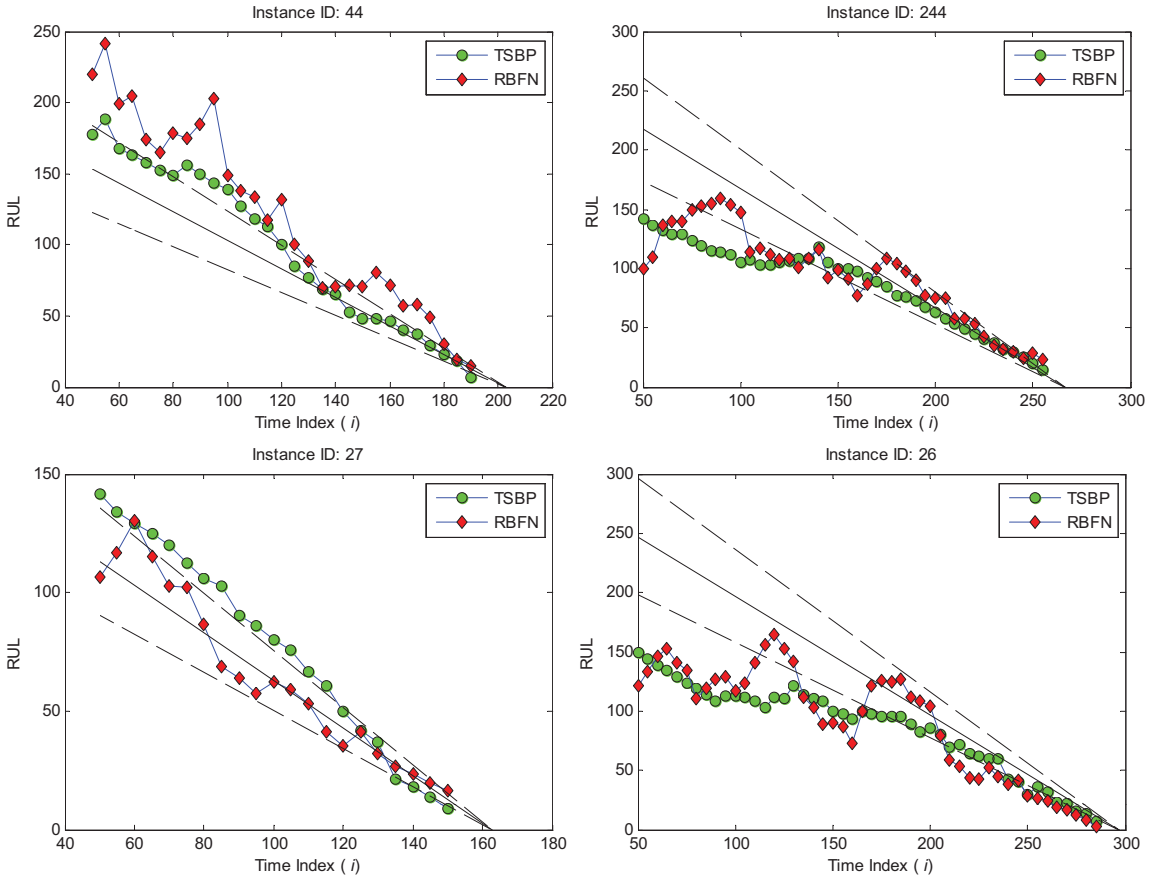

Figure 5.9: RUL predictions for selected instances using TSBP and RBFN method

- The performance of RBFN during final validation is worse than that during cross validation in this case.

5.4.3 Summary

In summary, using Neural Network to learn a global model from uncleaned data (e.g. with a mixture of diversified degradation patterns) may be difficult. The NN model is hard to converge to a small errors with a small amount of neurons. The NN model (with the same settings) trained from different training data set tends to be quite different. The performance of an NN is inconsistent during training (even if

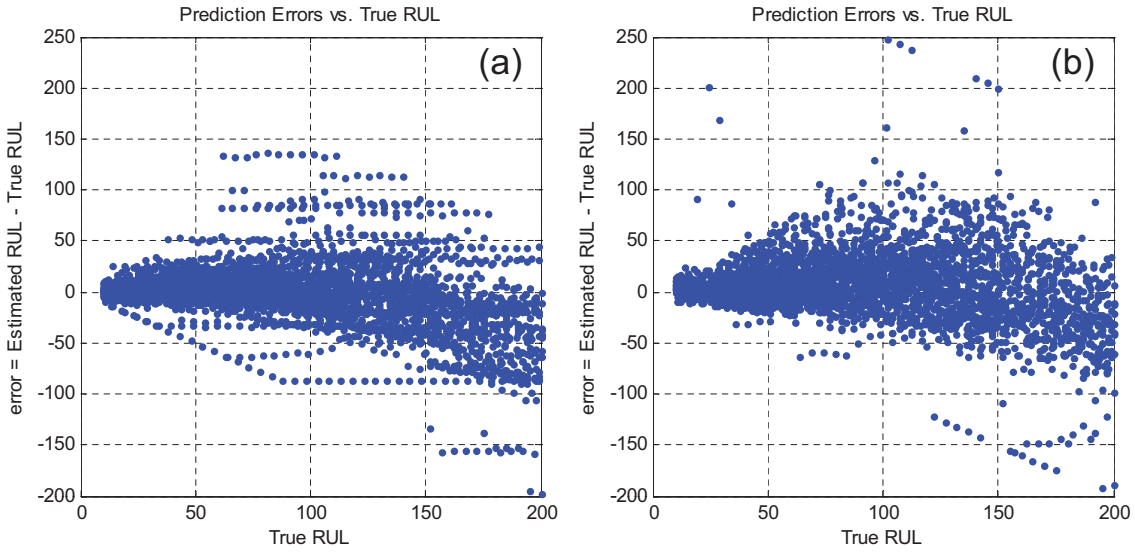

Figure 5.10: Prediction errors vs. True RUL for the final validation set, regardless time stamps and instances. (a) TSBP approach; (b) RBFN approach

cross-validation) and validation, which makes it hard to manage uncertainty of NN models, and also hard to develop confidence on the use of the NN model in a real application.

In this study, there are more training data for the final validation than the cross-validations. The increased amount of training data makes RBFN more difficult to reach the error objective set during cross-validation and more training data with higher diversity have worsened the performance of the algorithm. This may be attributed to the nature of a global modeling approach. An IBL based approaches such as TSBP, on the contrary, may see improved performance when the training data diversity increases.

The use of NN requires many settings and parameters being specified. Constructing the inputs to NN itself has endless options, not to mention the internal structure of NN. Therefore the performance of an NN model is highly dependent on the knowledge

and experience of the person who designs it. The NN model presented here is by no means optimized and can only be used as a reference.

5.5 Discussions

The turbofan engine degradation data demonstrates a number of special properties that pose challenges to RUL modeling.

Firstly, it is known that the data set contains two fault modes, even though they are not explicitly labeled. Global models will face difficulty in modeling such data without first classifying the data by fault modes. The information on fault modes, however, is not utilized while developing the TSBP model. In a TSBP model, mixture of fault modes in the data is actually treated as an unknown factor that influences the system's degradation process, and is handled naturally by the internal ensemble structure of TSBP. For a real-world system, the fault modes information is not always available for the collected data; some minor fault mechanism may have never be identified nor studied, and thus no diagnostic routine can be utilized to help label the fault modes. Therefore, a mixture of fault modes in the collected data can be pretty common in reality. This case study has been used to simulate such situation and test the RUL modeling capability.

Secondly, due to unknown engineering variance in difference instances, the collected data have shown quite large variance in the initial condition as well as degradation rate. This will increase the uncertainty in the model parameters learned from the data.

Lastly, due to unknown failure mechanism of the system (when the background knowledge to the data is not given), failure thresholds are hard to set based on the sensor measurements. Different instances have been claimed failure with quite large variance in the end-life sensor readings.

All these properties in the data can be observed from the normalized sensor plot shown in Figure 5.5. As an IBL approach, TSBP can naturally accommodate a variety of degradation patterns in the training instances when there's not enough knowledge or information to pre-classify them.

In this case study, run-to-failure condition data are available for the training instances, which tells the instances' actual life explicitly. However, run-to-failure data for the training instances are, though desirable, not mandatory. The TSBP approach is applicable as long as the training units' RUL can be estimated in a certain way, for example, through failure time distribution, heuristics, etc. The optimization method proposed by Tian et al. [2009] provides another means to deal with training instances that do not fail.

6 Conclusions

6.1 Summary

The degradation process of real-world systems is affected by many factors such as fault modes, usage patterns, initial engineering variance, etc., which may not be identifiable in an application. On the other hand, due to imperfect condition monitoring, certain parameters that affect the system's behavior, such as the operational and environmental conditions, may not be adequately measured as well. All these unmeasured or immeasurable factors will increase the seemingly noise and inconsistency in the collected data and pose great challenges to global models for RUL prediction. These challenges has been addressed by the proposed TSBP method.

In TSBP, the life-time condition data from past instances of a system with known failure time are used to extract a number of degradation patterns/trajectories through the kernel regression method and form a library of degradation models. For a test instance of the same type of system, similarity between it and each of the models in the library are evaluated by computing the minimal weighted Euclidean distances defined for two trajectories. Then a local RUL prediction for the test instance is given based on the known failure time of each of the degradations models. Finally the RUL probability density of the test instance can be estimated from the multiple local predictions using the kernel density estimation method.

Considering the specialty of engineering applications, three similarity definition is proposed in this thesis. The first definition is called Minimal Euclidean Distance with Time Lag (MED-TL), which looks for the best similarity by shifting the condition data in time and then computing the distance to the model trajectory. The second definition is called Minimal Euclidean Distance with Degradation Acceleration (MED-

DA), which allows the model trajectory to be scaled in time (either compressed or stretched) to accommodate degradation acceleration/deceleration. The third definition is called Minimal Euclidean Distance with Time Lag and Degradation Acceleration (MED-TL-DA), which includes both shifting and scaling operation while computing the distance between the trajectories.

A multi-regime data normalization method is developed to prepare data for TSBP modeling. The variables indicating the operational conditions of the system, if any, can be used to cluster the operation space into a number of operating regimes. Within each regimes, the operational conditions are relative stable and the extracted features can have comparable baseline. The features within each regime are normalized into a common range, so that they can be merged into a new time series with the original timestamps. The normalized features in the new time series are then weighted based on their relevance to the system degradation, which is evaluated by the proposed metric of Empirical Signal-Noise Ratio (eSNR). Finally the PCA method is applied to remove correlations among the normalized, weighted features.

In the case study, TSBP demonstrates effective prediction capability for complex system RUL estimation and noticeable improvement compared with the traditional Neural Network based prediction method.

6.2 Contributions and broader impacts

The key contributions of this thesis are summarized below:

1. Developed an effective RUL prediction method that addresses multiple challenges in complex system prognostics;
2. Derived three similarity metrics between degradation trajectories, which enrich the IBL methodology in prognostics applications;
3. Developed a multi-regime data normalization method for data preprocessing,

especially a variable weighting method applied as preprocessing procedure to the regular Principal Component Analysis.

In addition, a solid case study is provided in this thesis to fully explore the strength and weakness of the developed methodology.

The TSBP method developed in this thesis is expected to have a broad impact to industry. TSBP can be widely applied in engineering system prognostics applications where persistent data collection from a large number of identical machines or equipment are performed, such as for aircraft engines, heavy-duty mining trucks, wind farms and so on. Effective RUL predictions provided by the TSBP method will have great impact on the reduction of unplanned downtime and cost, safety assurance, and accomplishment of critical missions in such applications.

6.3 Comments on TSBP

The TSBP method presented in this chapter inherits the concept of IBL. It employs a similarity definition that relies on only the degradation trajectory – in other words, only the condition monitoring data – of the system, and thus requires minimal insight knowledge to the system. In principle, the TSBP approach will demonstrate its advantages when abundant historical data of similar instances are available, though theoretically the minimal number of training instances required is one. Therefore the TSBP approach will be more effective when persistent data collection from a large number of identical machines or equipment are performed, such as the application of aircraft engines, heavy-duty mining trucks, wind farms and so on.

TSBP has the nature of ensemble models, which make the method easily adapt to variations in the system's degradation pattern, especially mixture degradation patterns due to unknown factors. TSBP employs non-parametric representation of

degradation patterns, which requires no assumptions on the shape (e.g. exponential) of the degradation trend. TSBP makes RUL inference solely based on past cases with failure history, and thus requires no explicit failure criteria and requires minimal prior knowledge to the degradation mechanisms.

Although the TSBP approach does not mandate inside knowledge to the system, it does not prevent the inclusion of knowledge to improve the performance of the whole solution as well. Actually, the strength of an IBL approach is that it expresses the intuition in a simple way so that additional information or knowledge to the system can be easily incorporated. For example, the background information for the instances may help to narrow down the search space during instance retrieval and potentially improve the similarity evaluation accuracy. A diagnostic reasoner of the system, as well as a well-labeled training instances, can help the algorithm to make more focused predictions towards certain failure modes.

As an IBL approach, TSBP shares its "lazy learner" nature as well, which means TSBP will defer most of the computational task for learning to the evaluation stage. It is found that TSBP is very fast for model training, and there's no convergence issue during this stage; however, TSBP will have much higher computational load at evaluation stage. For instance, it may take a few seconds to make one RUL prediction on a regular mainstream PC; to generate a complete sequence of predictions at each time stamp of a test instance (e.g. 100 predictions) will take a few minutes; to make predictions for 100 test instances for validation purposes will take a few hours. As comparison, a trained NN can be at least 100 times faster during model evaluation, though the training processing of NN usually takes longer time than TSBP. The computation load of TSBP has prevented its parameters from being fully optimized through cross validation.

6.4 Future work

Some future work can be expected to further explore the capability of TSBP, improve the methodology and expand the RUL prediction framework brought forward by this thesis:

- Study the method to organize the degradation pattern library in order to retrieve fewer number of instances with high similarity during prediction;
- Improve the computational performance of distance evaluation of the current TSBP method;
- Explore other options of similarity/distance definitions;
- Benchmark TSBP with more other prediction techniques, especially ensemble techniques;
- More case studies on other engineering problems.

References

A. Aamodt and E. Plaza. Case-based reasoning - foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39–59, MAR 1994. ISSN 0921-7126.

M. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, FEB 2002. ISSN 1053-587X.

D. Banjevic and A. K. S. Jardine. Calculation of reliability function and remaining useful life for a Markov failure time process. *IMA J Management Math*, 17(2): 115–130, 2006. doi: 10.1093/imaman/dpi029.

P. Baruah and R. Chinnam. HMMs for diagnostics and prognostics in machining processes. *International Journal of Production Research*, 43(6):1275–1293, MAR 15 2005. ISSN 0020-7543. doi: {10.1080/00207540412331327727}.

J. Bi, K. Bennett, M. Embrechts, C. Breneman, and M. Song. Dimensionality reduction via sparse support vector machines. *J. Mach. Learn. Res.*, 3:1229–1243, 2003. ISSN 1532-4435.

P. Bonissone, A. Varma, and K. Aggour. A fuzzy instance-based model for predicting expected life: A locomotive application. In *Proceedings of the 2005 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications*, pages 20–25, 2005. ISBN 0-7803-9025-3. IEEE International Conference on Computational Intelligence for Measurement Systems and Applications, Messina, ITALY, JUL 20-22, 2005.

Z. I. Botev, J. F. Grotowski, and D. P. Kroese. Kernel density estimation via diffusion. *Annals of Statistics*, 2010. to be published.

G. Box, G. M. Jenkins, and G. Reinsel. *Time Series Analysis: Forecasting & Control (3rd Edition)*. Prentice Hall, 3rd edition, February 1994. ISBN 0130607746.

M. Carr and W. Wang. A case comparison of a proportional hazards model and a stochastic filter for condition-based maintenance applications using oil-based condition monitoring information. In *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, volume 222, page 4755, 2008. doi: {10.1243/1748006XJRR76}.

S. Cheng and M. Pecht. Multivariate state estimation technique for remaining useful life prediction of electronic products. In *AAAI Fall Symposium on Artificial Intelligence for Prognostics*, pages 26–32, November 2007. Arlington, VA.

R. Chinnam and P. Baruah. A neuro-fuzzy approach for estimating mean residual life in condition-based maintenance systems. *International Journal of Materials & Product Technology*, 20(1-3):166–179, 2004. ISSN 0268-1900.

R. B. Chinnam and P. Baruah. Autonomous diagnostics and prognostics in machining processes through competitive learning-driven HMM-based clustering. *International Journal of Production Research*, 47(23):6739–6758, 2009. ISSN 0020-7543. doi: {10.1080/00207540802232930}.

W. Cleveland. Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74(368):829–836, 1979. ISSN 0162-1459.

D. Cox. Regression models and life-tables. *Journal of the Royal Statistical Society Series B-Statistical Methodology*, 34(2):187–&, 1972. ISSN 1369-7412.

P. Craven and G. Wahba. Smoothing noisy data with spline functions - estimating the correct degree of smoothing by the method of generalized cross-validation. *Numerische Mathematik*, 31(4):377–403, 1979. ISSN 0029-599X.

M. Dong and D. He. A segmental hidden semi-Markov model (HSMM)-based diagnostics and prognostics framework and methodology. *Mechanical Systems and Signal Processing*, 21(5):2248–2266, JUL 2007. ISSN 0888-3270. doi: {10.1016/j.ymssp.2006.10.001}.

P. Frank. Fault-diagnosis in dynamic-systems using analytical and knowledge-based redundancy - a survey and some new results. *Automatica*, 26(3):459–474, MAY 1990. ISSN 0005-1098.

D. Frederick, J. DeCastro, and J. Litt. *Users Guide for the Commercial Modular Aero-Propulsion System Simulation (CMAPSS)*. NASA/ARL, 2007. Technical Manual TM2007-215026.

N. Gebraeel, M. Lawley, R. Liu, and V. Parmeshwaran. Residual life predictions from vibration-based degradation signals: A neural network approach. *IEEE Transactions on Industry Electronics*, 51(3):694–700, JUN 2004. ISSN 0278-0046. doi: {10.1109/TIE.2004.824875}.

K. Goebel, B. Saha, and S. A. A comparison of three data-driven techniques for prognostics. In *Failure prevention for system availability, 62th meeting of the MFPT Society*, page 119131, 2008. 62th meeting of the MFPT Society, Virginia Beach, VA, MAY 06-08, 2008.

I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.

M. A. Hall. *Correlation-based Feature Selection for Machine Learning*. PhD thesis, Department of Computer Science, The University of Waikato, Hamilton, NewZealand, April 1999.

F. O. Heimes. Recurrent neural networks for remaining useful life estimation. In *Proceedings of the 2008 International Conference on Prognostics and Health Management*, pages 1–6, oct. 2008. International Conference On Prognostics And Health Management, Denver, CO, Oct 06-09, 2008.

R. Huang, L. Xi, X. Li, C. R. Liu, H. Qiu, and J. Lee. Residual life predictions for ball bearings based on self-organizing map and back propagation neural network methods. *Mechanical Systems and Signal Processing*, 21(1):193–207, JAN 2007. ISSN 0888-3270. doi: {10.1016/j.ymssp.2005.11.008}.

J. Jang. ANFIS: Adaptive-Network-Based Fuzzy Inference System. *IEEE Transactions on Systems, Man, and Cybernetics*, 23:665–684, 1993.

A. Jardine, P. Anderso, and D. Mann. Application of the weibull proportional hazards model to aircraft and marine engine failure data. *Quality and Reliability Engineering International*, 3(2):77–82, DEC 1987. doi: {10.1002/qre.4680030204}.

A. K. S. Jardine, D. Lin, and D. Banjevic. A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical Systems and Signal Processing*, 20(7):1483–1510, OCT 2006. ISSN 0888-3270. doi: {10.1016/j.ymssp.2005.09.012}.

I. Jolliffe. *Pincipal Component Analysis*. New York: Springer, 2002.

R. Kohavi and G. H. John. Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2):273 – 324, 1997. ISSN 0004-3702. doi: 10.1016/S0004-3702(97)00043-X.

T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, August 2002. doi: 10.1109/5.58325.

I. J. Leontaritis and S. A. Billings. Input-output parametric models for non-linear systems part i: deterministic non-linear systems. *International Journal of Control*, 41(2):303–328, 1985. doi: 10.1080/0020718508961129.

R. Lewis and V. Torczon. A globally convergent augmented Lagrangian pattern search algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Optimization*, 12(4):1075–1089, APR 26 2002. ISSN 1052-6234.

Y. Li, S. Billington, C. Zhang, T. Kurfess, S. Danyluk, and S. Liang. Adaptive prognostics for rolling element bearing condition. *Mechanical Systems and Signal Processing*, 13(1):103–113, JAN 1999. ISSN 0888-3270.

Y. Li, T. Kurfess, and S. Liang. Stochastic prognostics for rolling element bearings. *Mechanical Systems and Signal Processing*, 14(5):747–762, SEP 2000. ISSN 0888-3270.

L. Liao and J. Lee. A novel method for machine performance degradation assessment based on fixed cycle features test. *Journal of Sound and Vibration*, 326(3-5):894–908, OCT 9 2009. ISSN 0022-460X. doi: {10.1016/j.jsv.2009.05.005}.

J. Liu, D. Djurdjanovic, J. Ni, N. Casoetto, and J. Lee. Similarity based method for manufacturing process performance prediction and diagnosis. *Computers in Industry*, 58(6):558–566, AUG 2007. ISSN 0166-3615. doi: {10.1016/j.compind.2006.12.004}.

J. MacGregor and T. Kourti. Statistical process-control of multivariate processes. *Control Engineering Practice*, 3(3):403–414, March 1995. ISSN 0967-0661. IFAC Symposium on Advanced Control of Chemical Processes (ADCHEM 94), KYOTO, JAPAN, MAY 25-27, 1994.

S. Marble and B. P. Morton. Predicting the remaining life of propulsion system bearings. In *2006 IEEE Aerospace Conference, Vols 1-9*, IEEE Aerospace Conference Proceedings, pages 4091–4098, 2006. ISBN 0-7803-9545-X. 2006 IEEE Aerospace Conference, Big Sky, MT, MAR 04-11, 2006.

G. Mclachlan and D. Peel. *Finite Mixture Models*. Wiley Series in Probability and Statistics. Wiley-Interscience, October 2000. ISBN 0471006262.

T. M. Mitchell. *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1 edition, March 1997. ISBN 0070428077.

R. M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996. ISBN 0387947248.

M. E. Orchard and G. J. Vachtsevanos. A particle-filtering approach for on-line fault diagnosis and failure prognosis. *Transactions of the Institute of Measurement and Control*, 31(3-4):221–246, JUN-AUG 2009. ISSN 0142-3312. doi: {10.1177/0142331208092026}.

L. R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proceedings of the IEEE*, pages 257–286, 1989.

C. Rasmussen. Gaussian processes in machine learning. In Bousquet, O and VonLuxburg, U and Ratsch, G, editor, *Advanced Lectures on Machine Learning*, volume 3176 of *Lecture Notes in Artificial Intelligence*, pages 63–71, 2004. ISBN 3-540-23122-6. Machine Learning Summer School Conference 2003, Tubingen, GERMANY, AUG, 2003.

A. Ray and S. Tangirala. Stochastic modeling of fatigue crack dynamics for on-line failure prognostics. *IEEE Transactions on Control Systems Technology*, 4(4):443–451, JUL 1996. ISSN 1063-6536.

S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2 edition, 2003.

B. Saha, K. Goebel, and J. Christophersen. Comparison of prognostic algorithms for estimating remaining useful life of batteries. *Transactions of the Institute of Measurement and Control*, 31(3-4):293–308, JUN-AUG 2009. ISSN 0142-3312. doi: {10.1177/0142331208092030}.

A. Saxena and K. Goebe. C-MAPSS Data Set, NASA Ames Prognostics Data Repository. http://ti.arc.nasa.gov/project/prognostic-data-repository, 2008.

A. Saxena, B. Wu, and G. Vachtsevanos. Integrated diagnosis and prognosis architecture for fleet vehicles using dynamic case-based reasoning. In *Autotestcon, 2005. IEEE*, pages 96 – 102, sept. 2005. doi: 10.1109/AUTEST.2005.1609109.

A. Saxena, J. Celaya, E. Balaban, K. Goebel, B. Saha, S. Saha, and M. Schwabacher. Metrics for evaluating performance of prognostic techniques. In *Proceedings of the 2008 International Conference on Prognostics and Health Management*, pages 1–17, oct. 2008a. doi: 10.1109/PHM.2008.4711436. International Conference On Prognostics And Health Management, Denver, CO, Oct 06-09, 2008.

A. Saxena, K. Goebel, D. Simon, and N. Eklund. Damage propagation modeling for aircraft engine run-to-failure simulation. In *Proceedings of the 2008 International Conference on Prognostics and Health Management*, pages 1–9, oct. 2008b. doi: 10.1109/PHM.2008.4711414. International Conference On Prognostics And Health Management, Denver, CO, Oct 06-09, 2008.

A. Saxena, J. Celaya, B. Saha, S. Saha, and K. Goebel. Metrics for offline evaluation of prognostic performance. *International Journal of Prognostics and Health Management*, April 2010. [Online] http://www.phmsociety.org/ijphm/2010/metrics-for-offline-evaluation-of-prognostic-performance.

M. Schwabacher and K. Goebel. A survey of artificial intelligence for prognostics. In *AAAI Fall Symposium*, pages 107–114, sept. 2007. Arlington, VA.

S. J. Sheather and M. C. Jones. A reliable data-based bandwidth selection method for kernel density estimation. *Journal of the Royal Statistical Society. Series B (Methodological)*, 53(3):683–690, 1991. ISSN 00359246.

P. SMYTH. Hidden markov-models for fault-detection in dynamic-systems. *Pattern Recognition*, 27(1):149–164, JAN 1994. ISSN 0031-3203.

M. Sugeno and T. Yasukawa. A fuzzy-logic-based approach to qualitative modeling. *Fuzzy Systems, IEEE Transactions on*, 1(1):7–31, February 1993. doi: 10.1109/TFUZZ.1993.390281.

T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to modeling and control. *IEEE transactions on systems, man and cybernetics*, 15(1): 116–132, 1985.

Z. Tian and M. J. Zuo. Health Condition Prognostics of Gears Using a Recurrent Neural Network Approach. In *Annual Reliability and Maintainability Symposium, 2009 Proceedings*, Reliability and Maintainability Symposium, pages 461–466, 2009. ISBN 978-1-4244-2508-2. 55th Annual Reliability & Maintainability Symposium, Ft Worth, TX, JAN 26-29, 2009.

Z. Tian, L. Wong, and N. Safaei. A neural network approach for remaining useful life prediction utilizing both failure and suspension histories. *Mechanical Systems and Signal Processing*, In Press, Corrected Proof:–, 2009. ISSN 0888-3270. doi: DOI:10.1016/j.ymssp.2009.11.005.

M. Tipping. Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1(3):211–244, SUM 2001. ISSN 1532-4435.

V. Torczon. On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, 7(1):1–25, FEB 1997. ISSN 1052-6234.

P. Tse and D. Atherton. Prediction of machine deterioration using vibration based fault trends and recurrent neural networks. *Journal of Vibration and Acoustics-Transactions of the ASME*, 121(3):355–362, JUL 1999. ISSN 1048-9002.

G. Vachtsevanos, F. LEWIS, M. Roemer, and A. Hess. *Intelligent fault diagnosis and prognosis for engineering systems*. John Wiley & Sons, Inc., 1 edition, 2006.

P. Vlok, J. Coetzee, D. Banjevic, A. Jardine, and V. Makis. Optimal component replacement decisions using vibration monitoring and the proportional-hazards model. *Journal of the Operational Research Society*, 53(2):193–202, FEB 2002. ISSN 0160-5682.

P. Vlok, M. Wnek, and M. Zygmunt. Utilising statistical residual life estimates of bearings to quantify the influence of preventive maintenance actions. *Mechanical Systems and Signal Processing*, 18(4):833–847, JUL 2004. ISSN 0888-3270. doi: {10.1016/j.ymssp.2003.09.003}.

A. Wald. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, 16(2):117–186, 1945. ISSN 00034851.

P. Wang and G. Vachtsevanos. Fault prognostics using dynamic wavelet neural networks. *AI EDAM-Artificial Intelligence for Engineering Design Analysis and Manufacturing*, 15(4):349–365, SEP 2001. ISSN 0890-0604.

T. Wang, J. Yu, D. Siegel, and J. Lee. A similarity-based prognostics approach for remaining useful life estimation of engineered systems. In *Proceedings of the 2008 International Conference on Prognostics and Health Management*, pages 1–6, oct. 2008. doi: {10.1109/PHM.2008.4711421}. International Conference On Prognostics And Health Management, Denver, CO, Oct 06-09, 2008.

W. Wang. A model to predict the residual life of rolling element bearings given monitored condition information to date. *IMA J Management Math*, 13(1):3–16, 2002. doi: 10.1093/imaman/13.1.3.

W. Wang and A. Christer. Towards a general condition based maintenance model for a stochastic dynamic system. *Journal of the Operational Research Society*, 51(2): 145–155, FEB 2000. ISSN 0160-5682.

W. Wang, P. Scarf, and M. Smith. On the application of a model of condition-based maintenance. *Journal of the Operational Research Society*, 51(11):1218–1227, NOV 2000. ISSN 0160-5682.

W. Wang, M. Golnaraghi, and F. Ismail. Prognosis of machine health condition using neuro-fuzzy systems. *Mechanical Systems and Signal Processing*, 18(4):813–831, JUL 2004. ISSN 0888-3270. doi: {10.1016/S0888-3270(03)00079-7}.

S. Wegerich. Similarity based modeling of time synchronous averaged vibration signals for machinery health monitoring. In *2004 IEEE Aerospace Conference Proceedings, Vols 1-6*, IEEE Aerospace Conference Proceedings, pages 3654–3662, 2004. ISBN 0-7803-8155-6. IEEE Aerospace Conference, Big Sky, MT, MAR 06-13, 2004.

G. Welch and G. Bishop. An introduction to the kalman filter. Technical report, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 1995.

G. A. Whitmore. First-passage-time models for duration data: Regression structures and competing risks. *Journal of the Royal Statistical Society. Series D (The Statistician)*, 35(2):207–219, 1986. ISSN 00390526.

C. Williams. Prediction with Gaussian processes: From linear regression to linear prediction and beyond. In Jordan, MI, editor, *Learning in Graphical Models*, volume 89 of *NATO Advanced Science Institutes Series, Series D, Behavioral and Social Sciences*, pages 599–621, 1998. ISBN 0-7923-5017-0. NATO Advanced Study Institute on Learning in Graphical Models, ERICE, ITALY, SEP 27-OCT 07, 1996.

F. Xue, P. Bonissone, A. Varma, W. Yan, N. Eklund, and K. Goebel. An Instance-Based Method for Remaining Useful Life Estimation for Aircraft Engines. *Journal of Failure Analysis and Prevention*, 8(2):199–206, 2008. doi: {10.1007/s11668-008-9118-9}.

R. Yam, P. Tse, L. Li, and P. Tu. Intelligent predictive decision support system for condition-based maintenance. *International Journal of Advanced Manufacturing Technology*, 17(5):383–391, 2001. ISSN 0268-3768.

J. Yan, M. Koc, and J. Lee. A prognostic algorithm for machine performance assessment and its application. *Production Planning & Control*, 15(8):796–801, DEC 2004. ISSN 0953-7287. doi: {10.1080/09537280412331309208}.

E. Zio and F. D. Maio. A data-driven fuzzy approach for predicting the remaining useful life in dynamic failure scenarios of a nuclear system. *Reliability Engineering & System Safety*, 95(1):49 – 57, 2010. ISSN 0951-8320. doi: DOI:10.1016/j.ress.2009.08.001.

Appendix: PCA with Karhunen-Loéve transform

Let $X = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_L)$ be a matrix consisting of L samples for the N-dimensional variable $\mathbf{x}$, with each column vector being one sample. The empirical mean of $\mathbf{x}$ is given as

$$\mathbf{u} = (\mu_1, \mu_2, ..., \mu_N)^T$$
$$\mu_n = \frac{1}{L}\sum_{l=1}^{L} X[n, l] \tag{A.1}$$

By removing the empirical mean, we have

$$Y = X - \mathbf{uh} \tag{A.2}$$

where $\mathbf{h}$ is a $1 \times N$ vector of all 1's.

Then the empirical covariance matrix of Y is computed:

$$C = \frac{1}{L-1} Y \cdot Y^T \tag{A.3}$$

The $N \times N$ empirical covariance C can be decomposed by the eigenvalues and eigenvectors

$$C = V \cdot \Lambda \cdot V^T \tag{A.4}$$

where Λ is an $N \times N$ diagonal matrix of the eigenvalues of C, with $\Lambda[n, n] = \lambda_n$; and $V = (\mathbf{v}_1, ..., \mathbf{v}_N)$ is an $N \times N$ matrix of the eigenvectors, with each column vector $\mathbf{v}_n$ being an eigenvector corresponding to the eigenvalue λ_n. In matrix C, the eigenvectors λ_n is sorted in ascending order such that $\lambda_n >= \lambda_{n+1}$ for $n = 1, ..., N-1$.

Next, a subset of the eigenvectors will be selected. The first M eigenvectors are selected so that the first M eigenvalues sum up no less than a certain percentage, say 90%, of the total sum of all the eignenvalues:

$$M = \min_{m=1,...,N} m$$
$$\sum_{n=1}^{m} \lambda_n \geq 90\% \times \sum_{n=1}^{N} \lambda_n \tag{A.5}$$

The selected M eigenvectors will form an $N \times M$ matrix

$$V_M = (\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_M) \tag{A.6}$$

Now, the samples in X can be transformed into the principal component domain with transformation

$$Z = V_M^T \cdot Y = V_M^T \cdot (X - \mathbf{uh}) \tag{A.7}$$

where Z is an $M \times L$ matrix with each column being one principal component vector for the corresponding column vector in X.

For a new sample of $\mathbf{x}$ that is not from samples in X, it can also be transformed using the parameters established from X:

$$\mathbf{z} = V_M^T \cdot (\mathbf{x} - \mathbf{u}) \tag{A.8}$$

The resultant principal component vector z is M-dimensional.